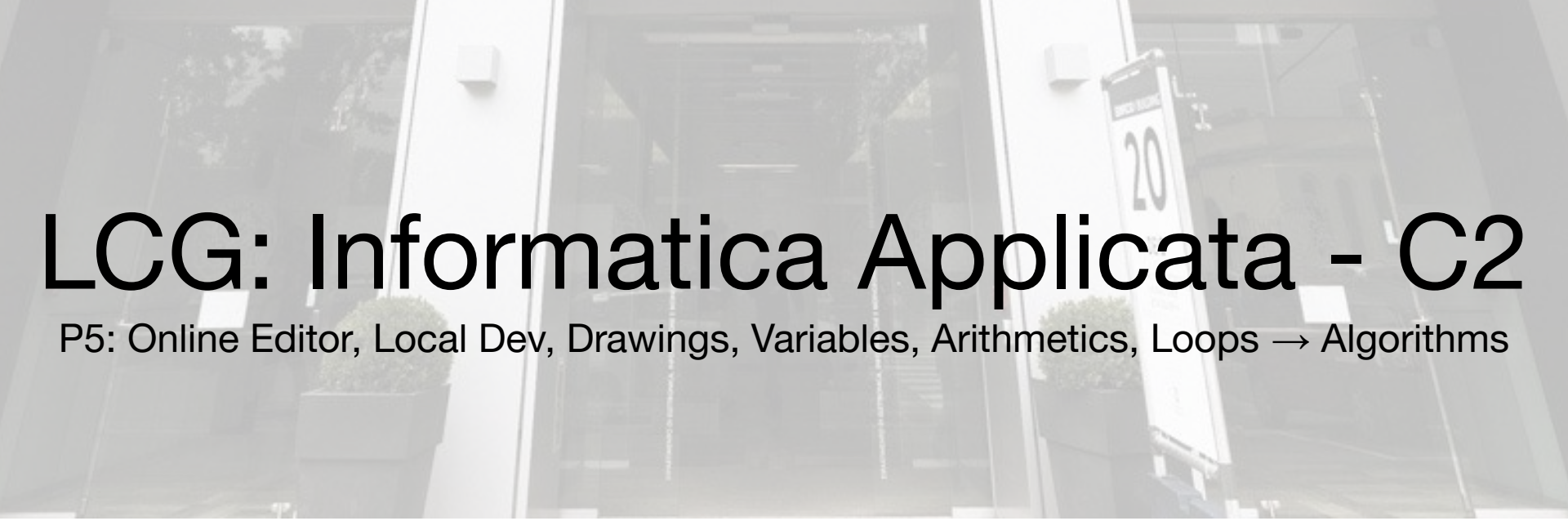


LCG: Informatica Applicata - C2

P5: Online Editor, Local Dev, Drawings, Variables, Arithmetics, Loops → Algorithms

Scuola del Design
Laurea Triennale in Communication Design

Aula B6.3.1, Politecnico di Milano
October 1st, 2024

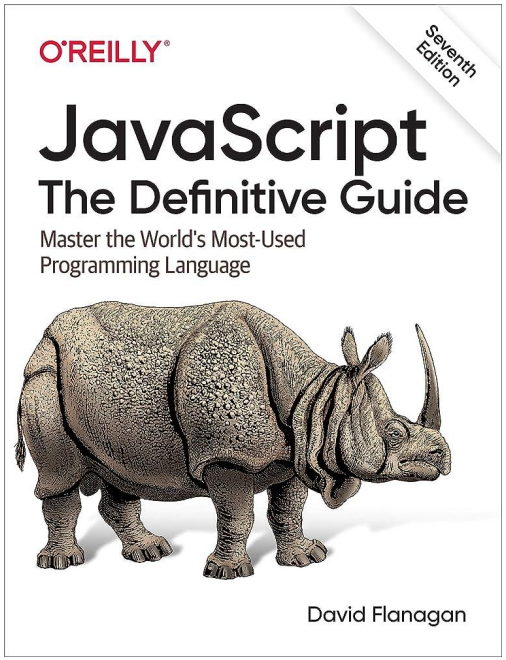
Davide Conficconi <davide.conficconi@polimi.it>
Alessandro Nazzari <alessandro.nazzari@polimi.it>

POLITECNICO
MILANO 1863

Important things: **Material**

https://bit.ly/lcg-c2-2024

https://webeep.polimi.it/course/view.php?id=307969



https://www.w3schools.com/js/default.asp

- https://git-scm.com/doc
  https://docs.github.com/en/get-started/using-git/about-git
- https://docs.github.com/en/pages/quickstart
- https://docs.github.com/en/pages/getting-started-with-github-p
  /creating-a-github-pages-site
- https://www.w3schools.com/git/git_intro.asp?remote=github
- https://git-scm.com/book/en/v2
- Github pages Docs
- https://docs.github.com/en/pages/getting-started-with-github-pages
  /creating-a-github-pages-site#next-steps
- Bootstrap reference
- From markdown to github pages

Recall: Last Time Objectives

1. ~~Informatica Self-Assesment~~
2. ~~Local dev with VSCode, Indice html for IA2425~~
3. ~~Create our Webpage with Githubpages~~
4. Draw with P5, P5 Live Editor

https://editor.p5js.org/

Recall: Informatica Self-Assesment

https://forms.office.com/e/40z2YF0ZnA

Image by Sergio Cerrato - Italia from Pixabay

Recall: Definition of Computer Science

Science of

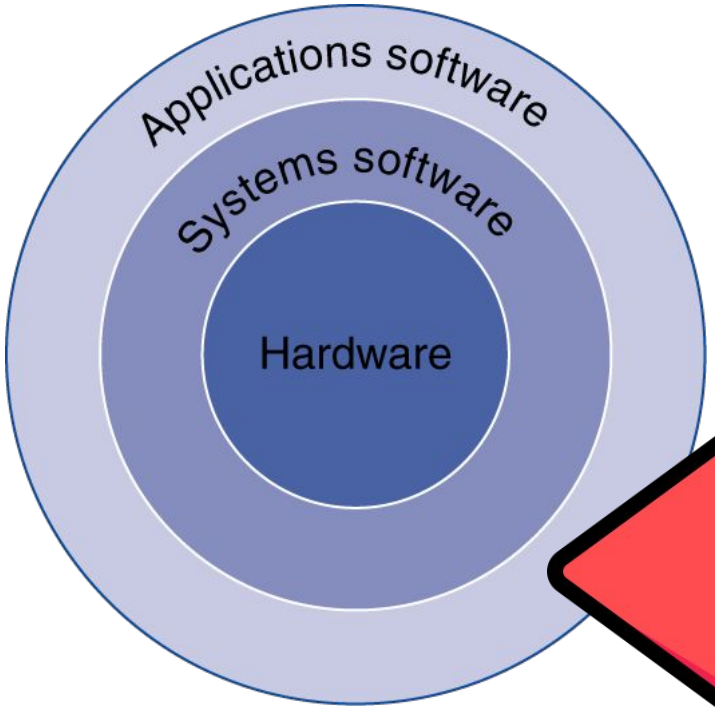
representation

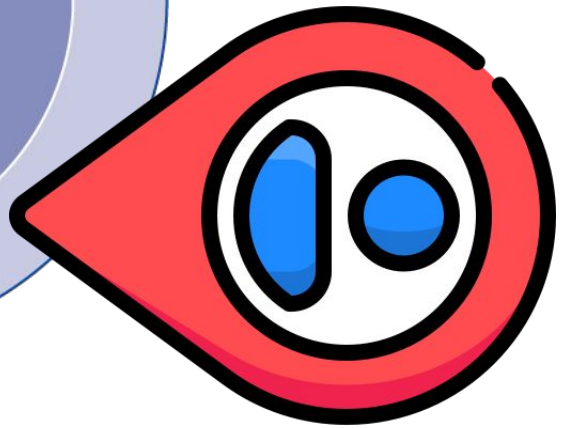
and

elaboration

of **information**

Recall: Abstraction Stack of Reference





Recall: Visual Studio Code

Visual Studio Code (VSCode) è l'ambiente di sviluppo di riferimento per il nostro corso

Si tratta di un software liberamente installabile da:

https://code.visualstudio.com/download

I repository/market del proprio sistema operativo

Dopo aver installato VSCode, avviamolo e procediamo con l'installazione dell'estensione per lo sviluppo locale

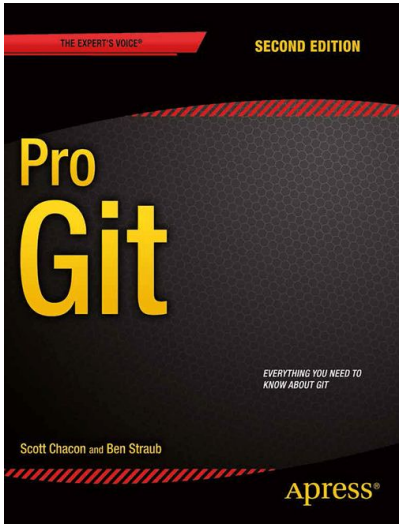
Recall: HTML: HyperText Markup Language

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>Informatica Applicata</title>
</head>
<body>
    <h1>Esempio di pagina HTML</h1>
    <p align="center">Questa è una banale pagina HTML </p>
</body>
</html>
```

Recall: HTML: HyperText Markup Language

https://www.w3schools.com/html/default.asp

Recall:



Single developer

- Tracks evolution
- Builds history
- Navigate the history

Team of developers

- Allow concurrent development
- Track the responsible
- Support in merging changes

https://git-scm.com/book/en/v2

Recall: To Build our “web portfolio” of LCG

Indice di Informatica Applicata C2

Terremo traccia degli sviluppi tramite questo indice.

- Lezione 1
- Lezione 2
- Lezione 3
- Lezione 4
- Lezione 5
- Lezione 6

https://davideconficconi.github.io/indice-lcg2425/index.html

https://pages.github.com/

https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site#next-steps

Recall: Further *readings*

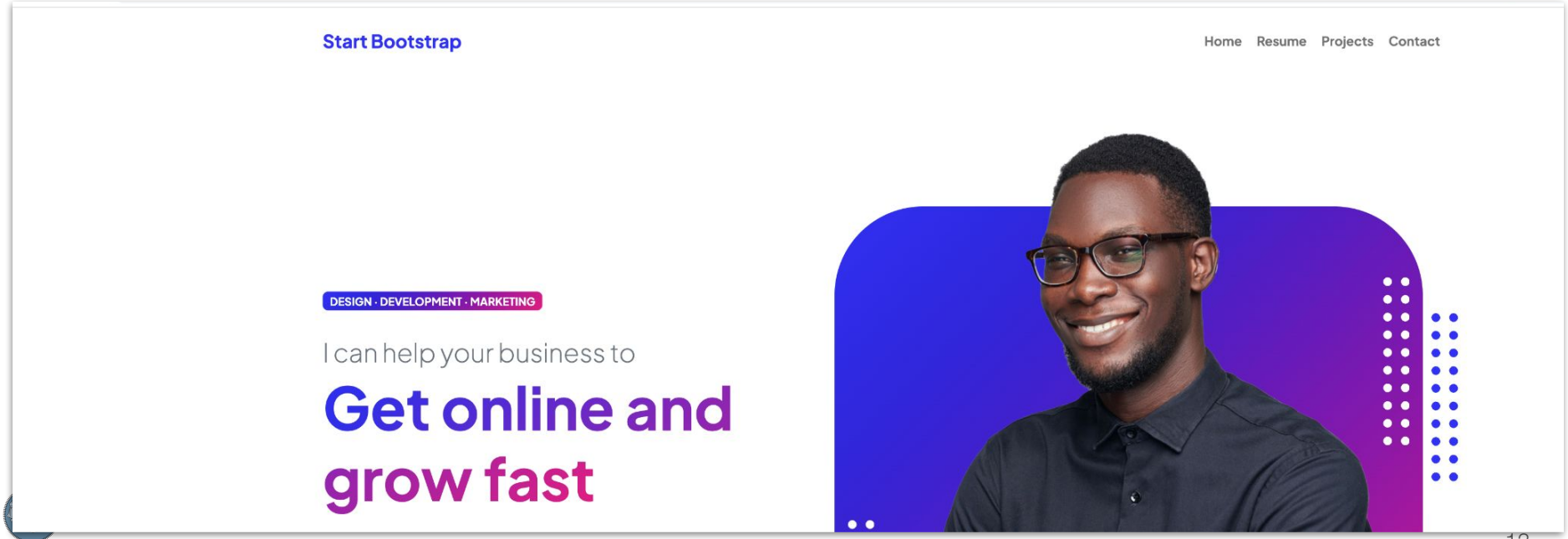
1) https://git-scm.com/book/en/v2
2) Github pages Docs
3) https://docs.github.com/en/pages/getting-started-with-github-pages/creating-a-github-pages-site#next-steps
4) Bootstrap reference
5) From markdown to github pages
6) https://thecodingtrain.com/tracks/git-and-github-for-poets

Recall: Lecture 1 Challenge

Prepare a better index visualization than the one with plain html I presented.

You might use your CSS knowledge or bootstrap templates

Deadline: Monday 30th September 20:00



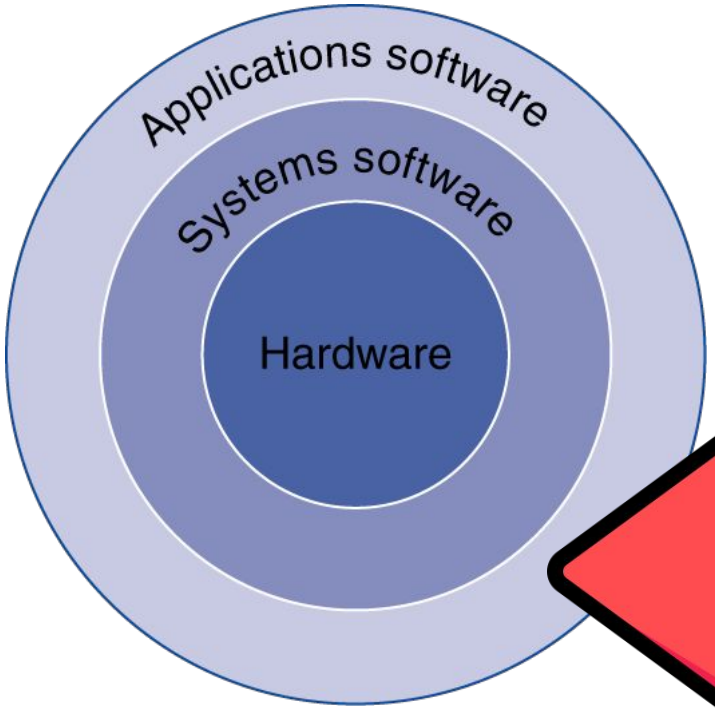
Tempo dell'Appello!

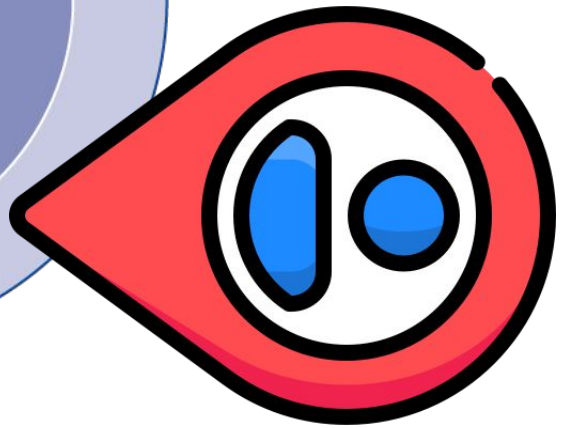
Today Objectives

1. ~~Recap + Challenge discussion~~
2. Refresh on Algorithms
3. Dev with P5: Live Editor and Local VScode
4. Draw with P5
5. Variables + JS arithmetic
6. Repetitions
7. Branches
8. Put everything live and check github!

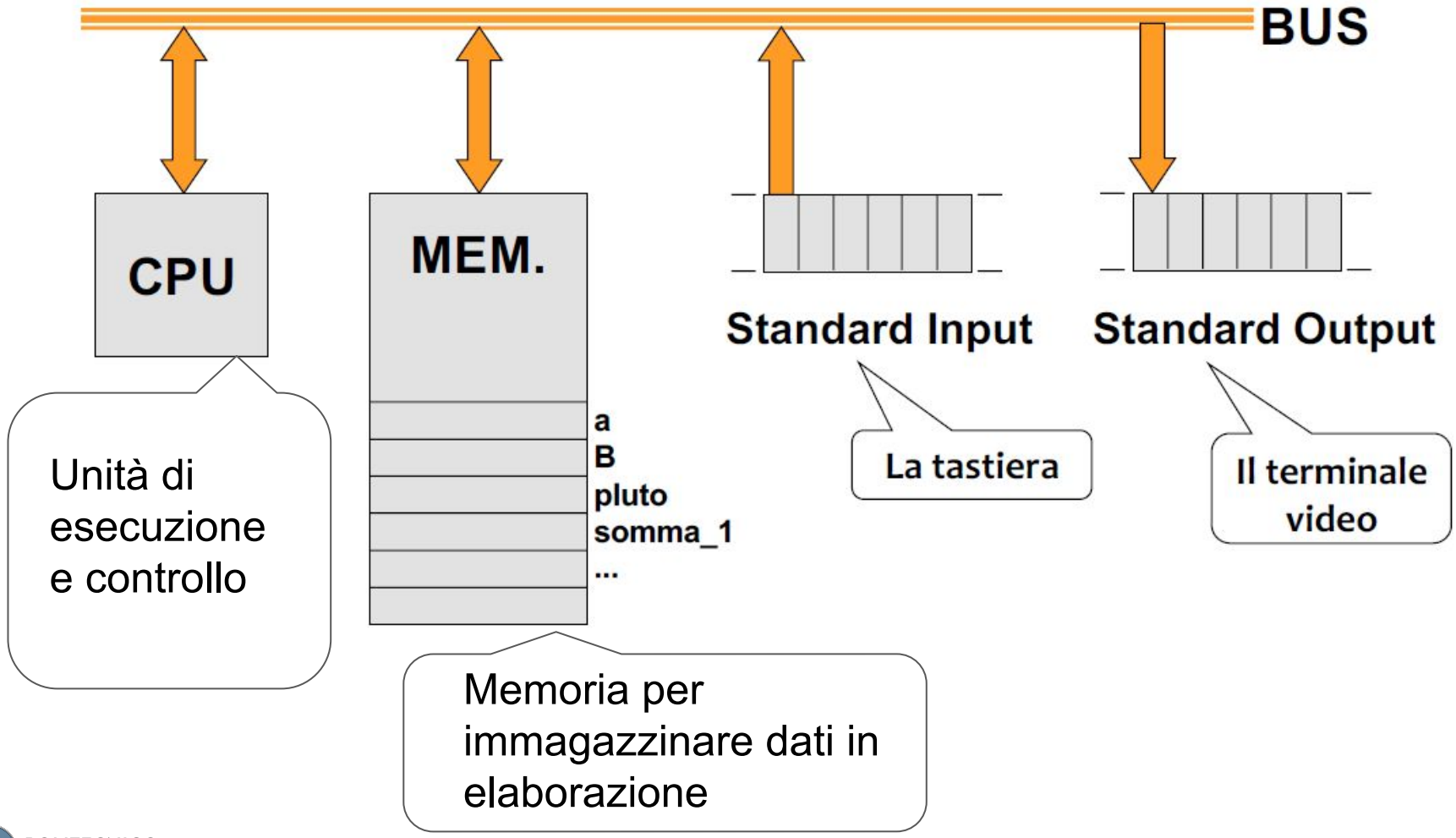
https://editor.p5js.org/

Recall: Abstraction Stack of Reference





Recall: Il “calcolatore” astratto



Una definizione di Algoritmo

Una definizione di Algoritmo

"Una sequenza di passi, definiti con precisione, che portano alla realizzazione di un compito."[1]

Una definizione di Algoritmo

"Una sequenza di passi, definiti con precisione, che portano alla realizzazione di un compito."[1]

Come si specifica un algoritmo?

Una definizione di Algoritmo

"Una sequenza di passi, definiti con precisione, che portano alla realizzazione di un compito."[1]

Come si specifica un algoritmo?

Utilizzando dello pseudocodice

Se A > B **allora** B = A **altrimenti** A = B

Utilizzando dei diagrammi di flusso (aka schemi a blocchi)

Una definizione di Algoritmo

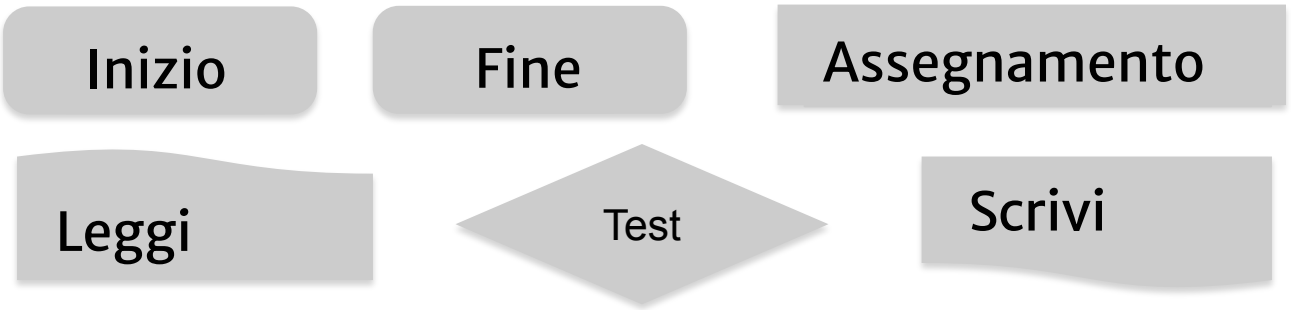
"Una sequenza di passi, definiti con precisione, che portano alla realizzazione di un compito."[1]

Come si specifica un algoritmo?

Utilizzando dello pseudocodice

Se A > B **allora** B = A **altrimenti** A = B

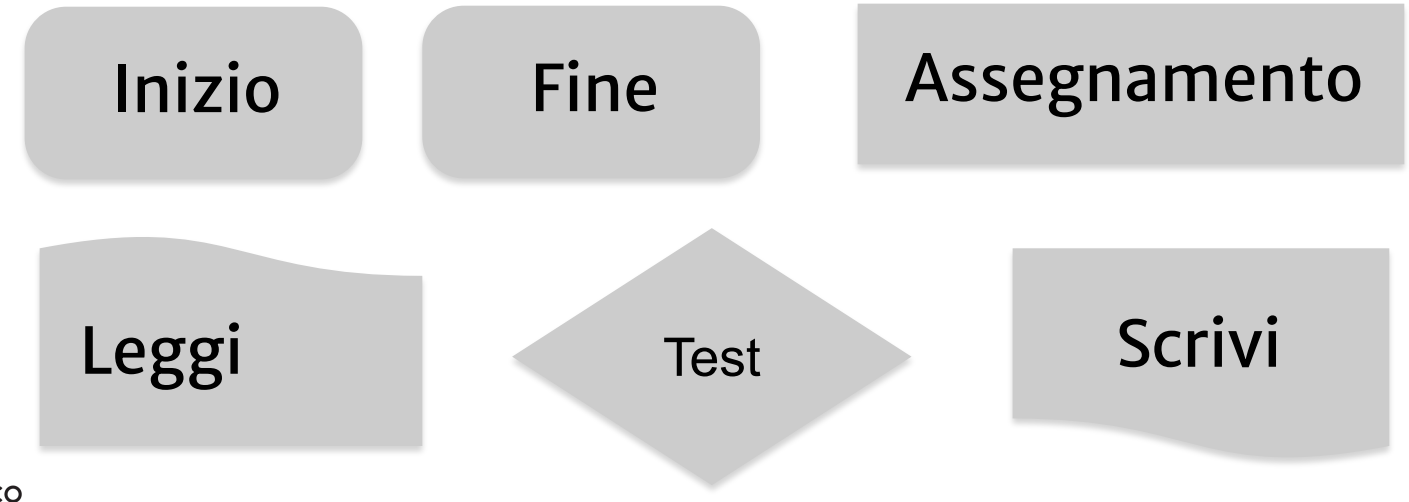
Utilizzando dei diagrammi di flusso (aka schemi a blocchi)



Esempio1: Perimetro e Area di Circonf.

Si realizzi il diagramma di flusso dell'algoritmo che **acquisisce** un valore intero positivo (è senz'altro così) che rappresenta il **raggio** di una **circonferenza** e calcola e visualizza il **perimetro** a e **l'area.**

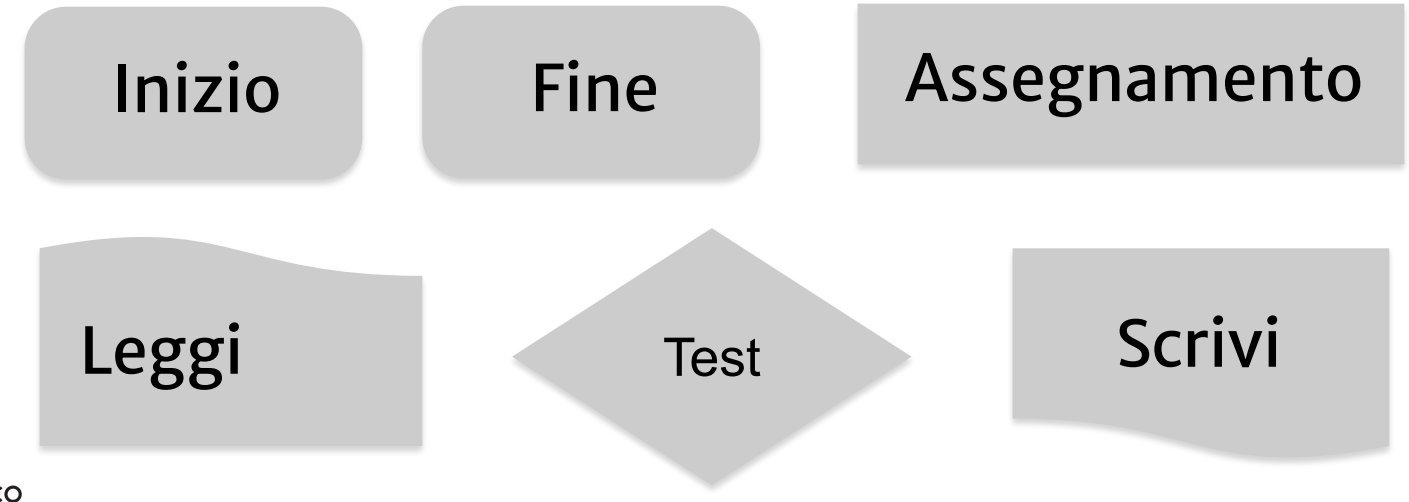
I nostri strumenti:



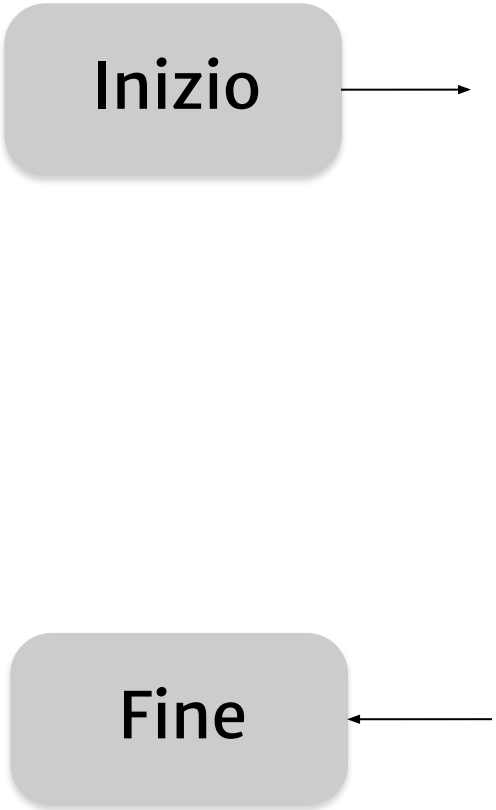
Esempio1: Perimetro e Area di Circonf.

Si realizzi il diagramma di flusso dell'algoritmo che **acquisisce** un valore intero positivo (è senz'altro così) che rappresenta il **raggio** di una **circonferenza** e calcola e visualizza il **perimetro (*P=2πr*)** a e **l'area (*A=πr²*)**.

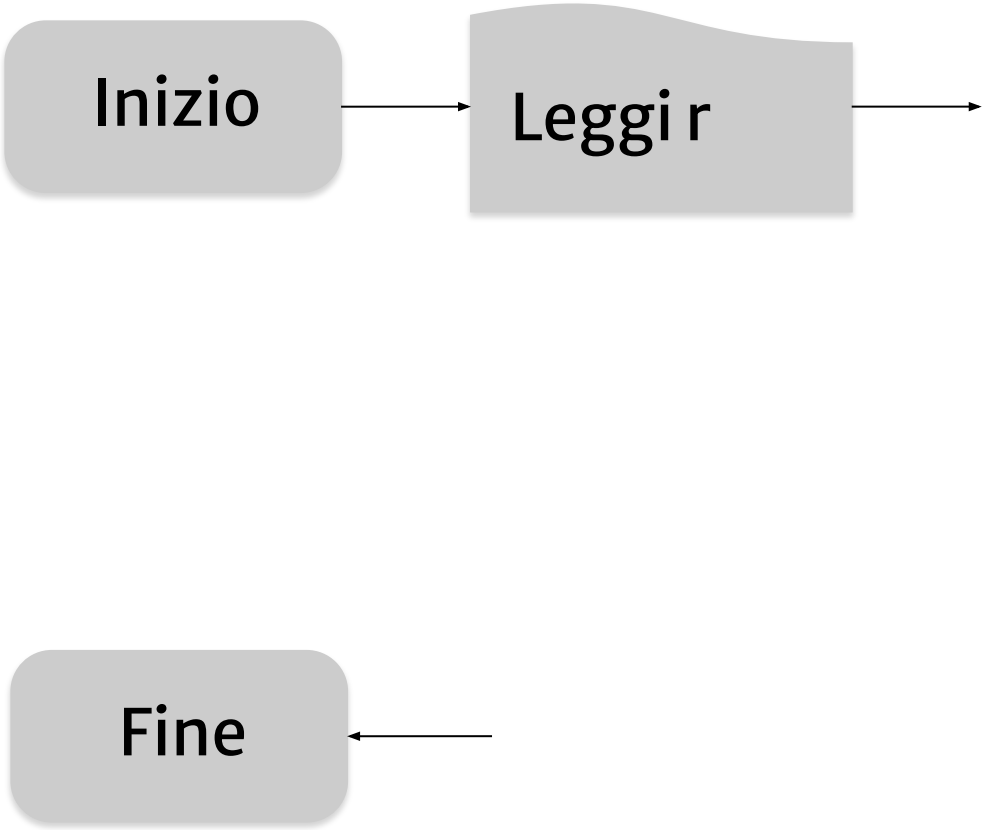
I nostri strumenti:



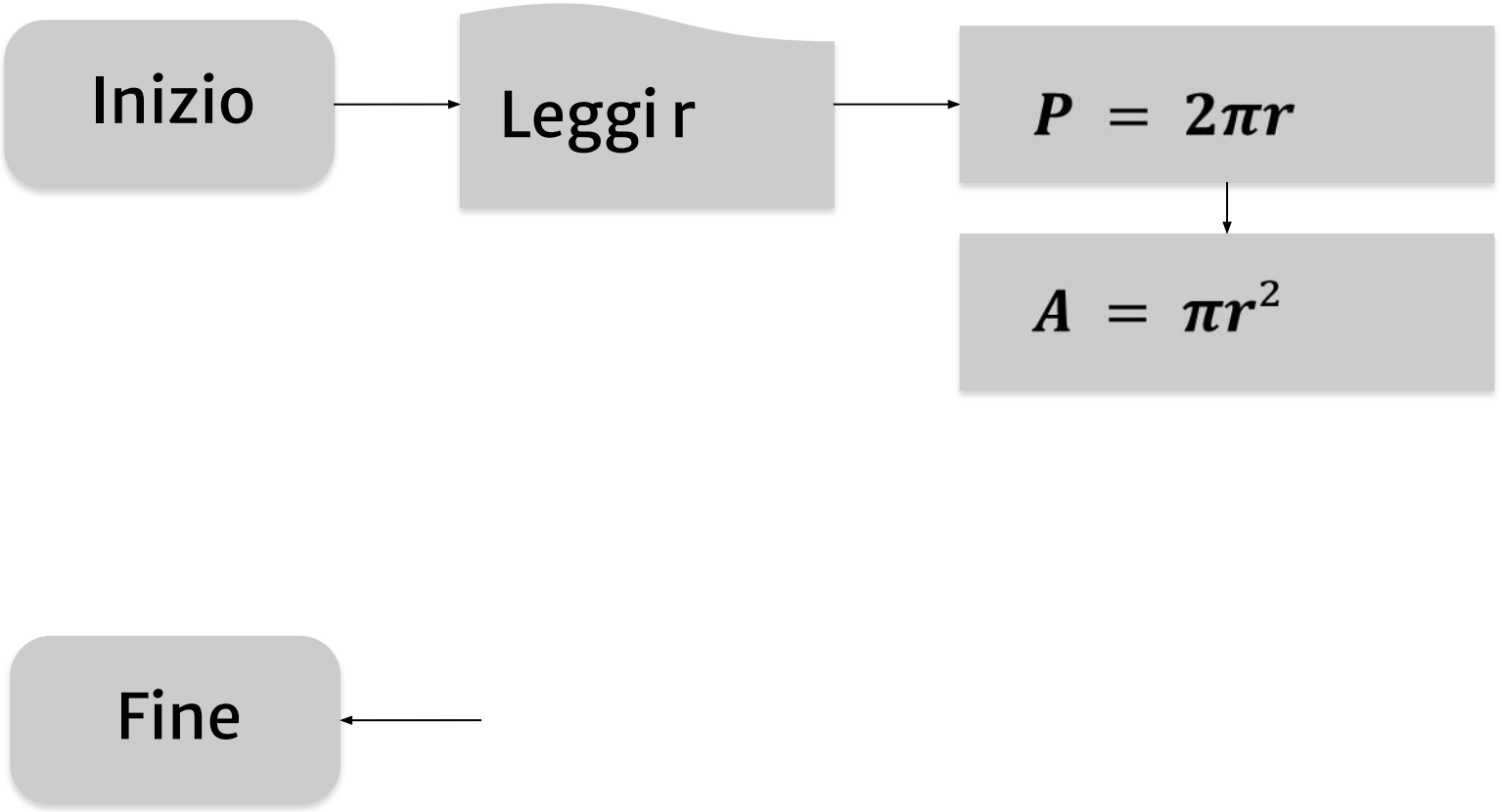
Soluzione: Perimetro e Area di Circonf.



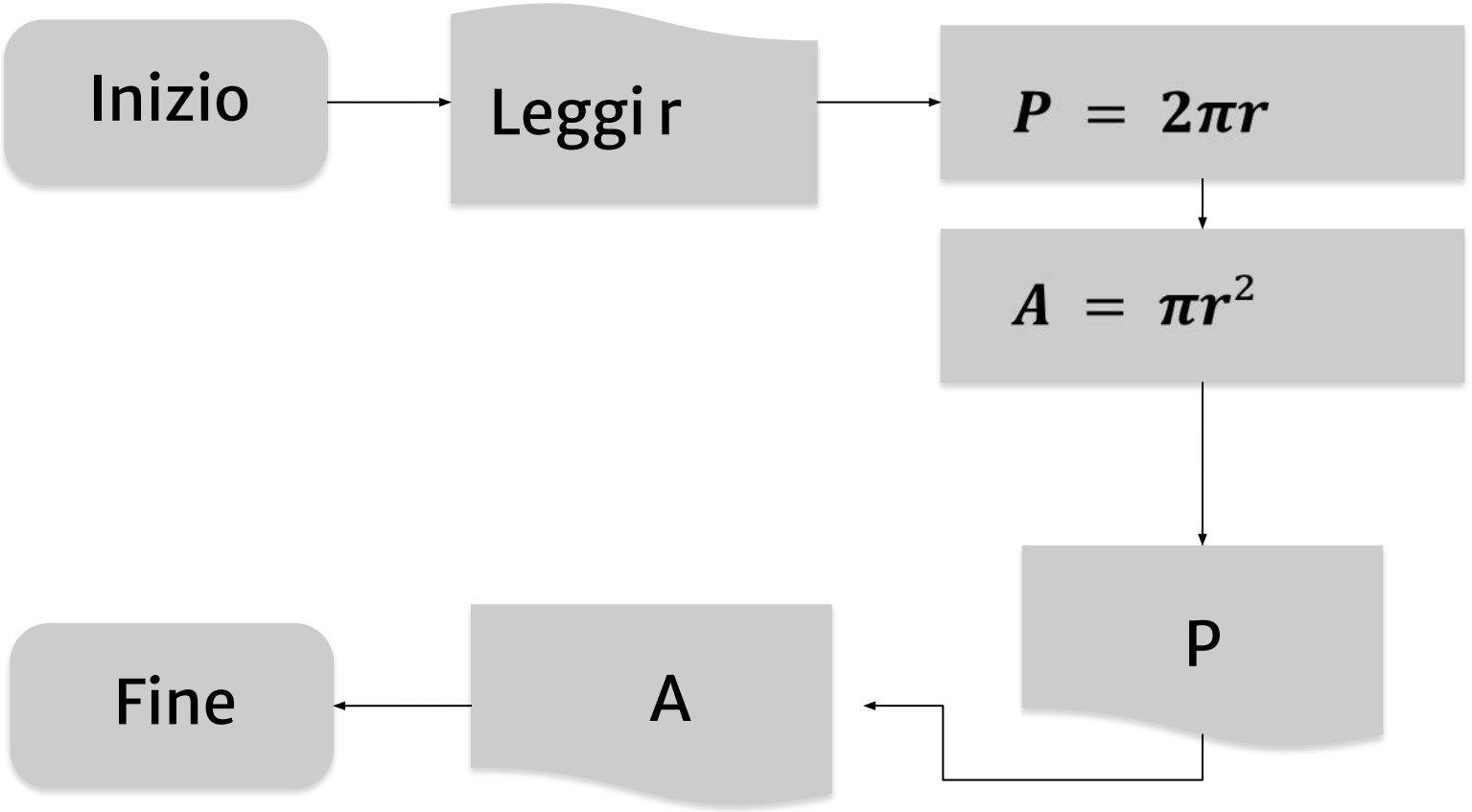
Soluzione: Perimetro e Area di Circonf.



Soluzione: Perimetro e Area di Circonf.



Soluzione: Perimetro e Area di Circonf.



Today Objectives

1. ~~Recap + Challenge discussion~~
2. ~~Refresh on Algorithms~~
3. Dev with P5: Live Editor and Local VScode
4. Draw with P5
5. Variables + JS arithmetic
6. Repetitions
7. Branches
8. Put everything live and check github!

https://editor.p5js.org/

Cool…and now?

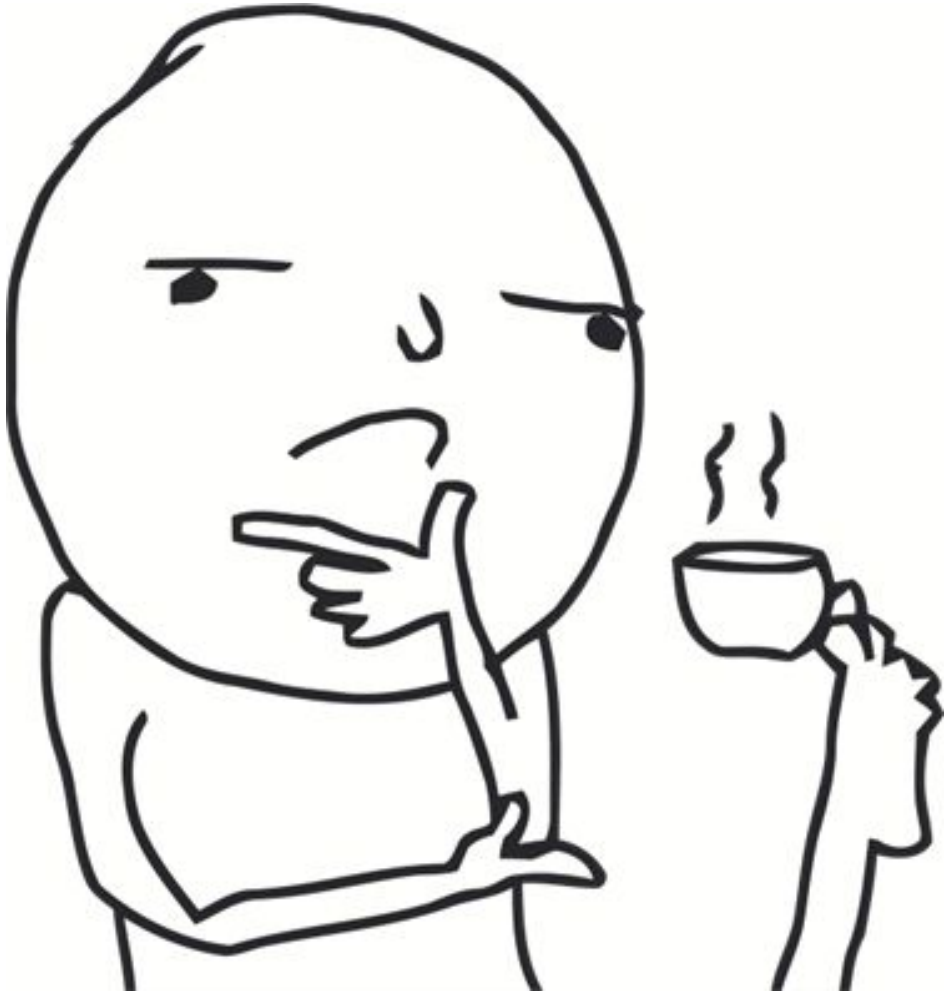


Image by https://knowyourmeme.com/memes/thinking-foot-thinking-holding-cup-foot

JavaScript (JS)

Chi sviluppa software non deve necessariamente imparare a memoria tutte le funzioni e strutture dati di un linguaggio

La documentazione online (References) ci permette di andare a rivedere nello specifico lo scopo e la sintassi dei costrutti di nostro interesse
https://developer.mozilla.org/en-US/docs/Web/JavaScript

What is P5? https://hello.p5js.org/

What is P5? https://hello.p5js.org/

What is P5? https://hello.p5js.org/

p5.js is a **friendly** tool for
learning to code and make art.

Why p5.js?

Make **programming** interactive graphics **easy**

Why p5.js?

Make **programming** interactive graphics **easy**

```
const width = 928; const height = 500; const marginTop =
const marginRight = 0; const marginBottom = 30; const marginLef
// Declare the x (horizontal position) scale.
const x = d3.scaleBand()
    .domain(d3.groupSort(new_data, ([d]) => -d.frequency, (d) =>
d.letter)) // descending frequency
    .range([marginLeft, width - marginRight])
    .padding(0.1);
// Declare the y (vertical position) scale.
const y = d3.scaleLinear()
    .domain([0, d3.max(new_data, (d) => d.frequency)])
    .range([height - marginBottom, marginTop]);
// Create the SVG container.
const svg = d3.create("svg").attr("width", width).attr("height",
height)
    .attr("viewBox", [0, 0, width, height])
    .attr("style", "max-width: 100%; height: auto;");
// Add a rect for each bar.
svg.append("g").attr("fill", "steelblue")
 .selectAll().data(new_data).join("rect")
    .attr("x", (d) => x(d.letter)).attr("y", (d) => y(d.frequency))
    .attr("height", (d) => y(0) - y(d.frequency))
    .attr("width", x.bandwidth());
// Add the x-axis and label.
svg.append("g")
    .attr("transform", `translate(0,${height - marginBottom})`)
    .call(d3.axisBottom(x).tickSizeOuter(0));
// Add the y-axis and label, and remove the domain line.
svg.append("g").attr("transform", `translate(${marginLeft},0)`)
    .call(d3.axisLeft(y).tickFormat((y) => (y * 100).toFixed()))
    .call(g => g.select(".domain").remove())
    .call(g => g.append("text")
        .attr("x", -marginLeft).attr("y", 20)
        .attr("fill", "currentColor").attr("text-anchor", "start")
        .attr("font-size","20").text("↑ Frequency (%)"));

svg.selectAll(".tick text")
  .attr("font-size","20");
```

Why p5.js?

Make **programming** interactive graphics **easy**

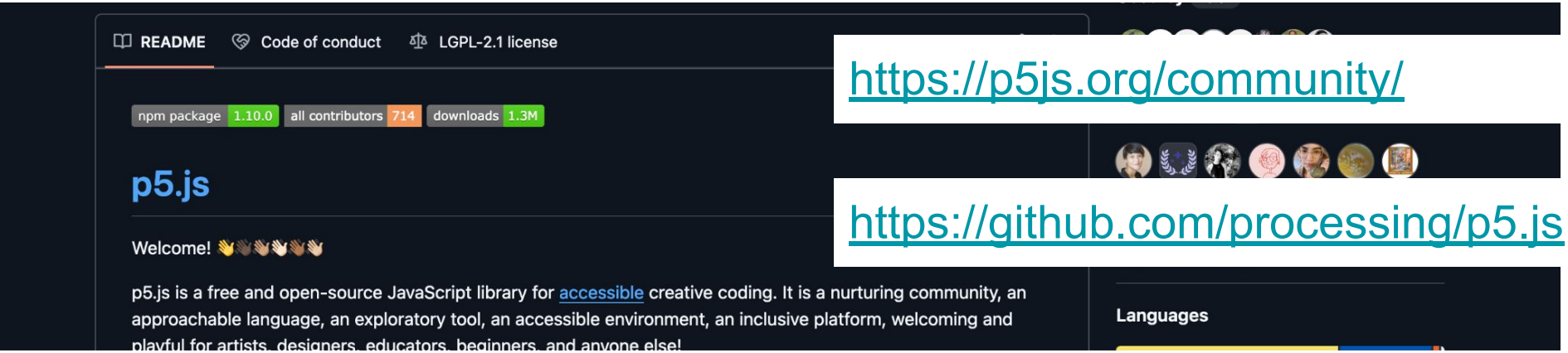
Immediate feedback with few lines of code

Why p5.js?

Make **programming** interactive graphics **easy**

Immediate feedback with few lines of code

It is a **free** and **open-source** JavaScript library built by an inclusive, nurturing **community**. p5.js welcomes artists, designers, beginners, educators, and anyone else!

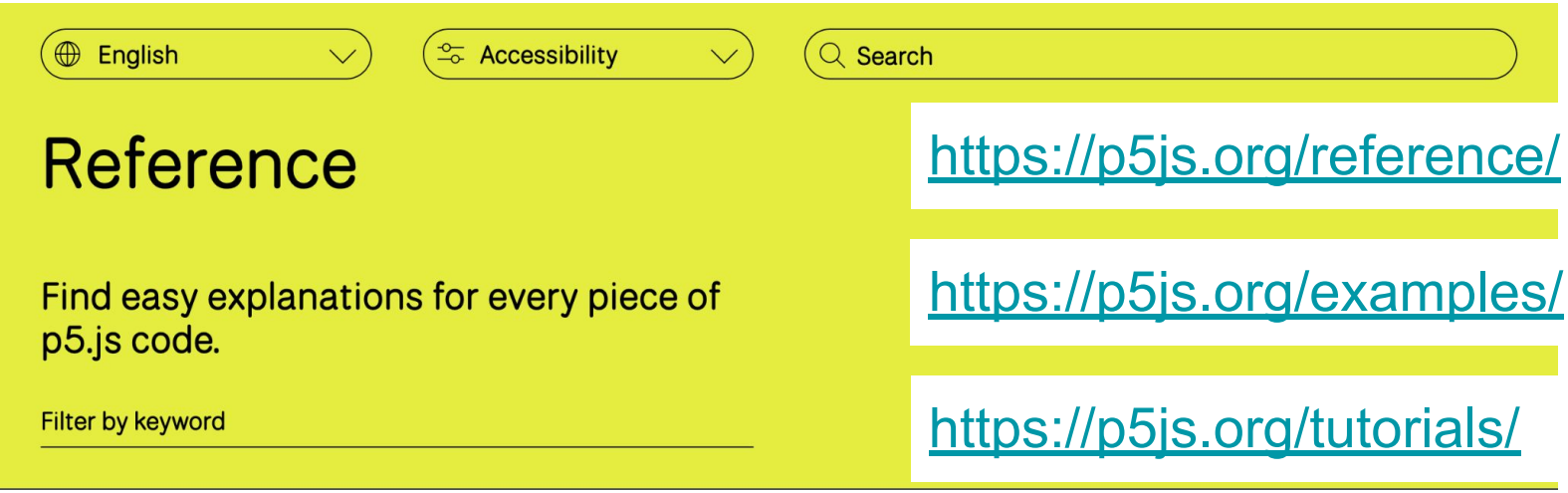


https://p5js.org/community/

https://github.com/processing/p5.js

Why p5.js?

Make **programming** interactive graphics **easy**



The Most Basic p5.js sketch

The Most Basic p5.js sketch

```
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(220);
}
```

The Most Basic p5.js sketch

setup() is called and runs one time. It can be used to set default values for your project.

```
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(220);
}
```

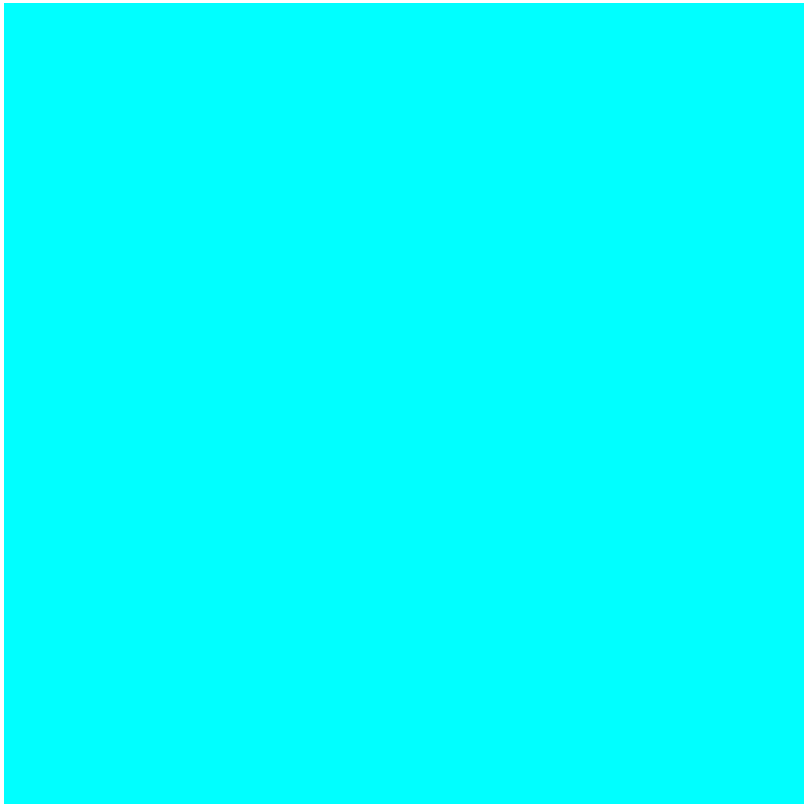
The Most Basic p5.js sketch

setup() is called and runs one time. It can be used to set default values for your project.

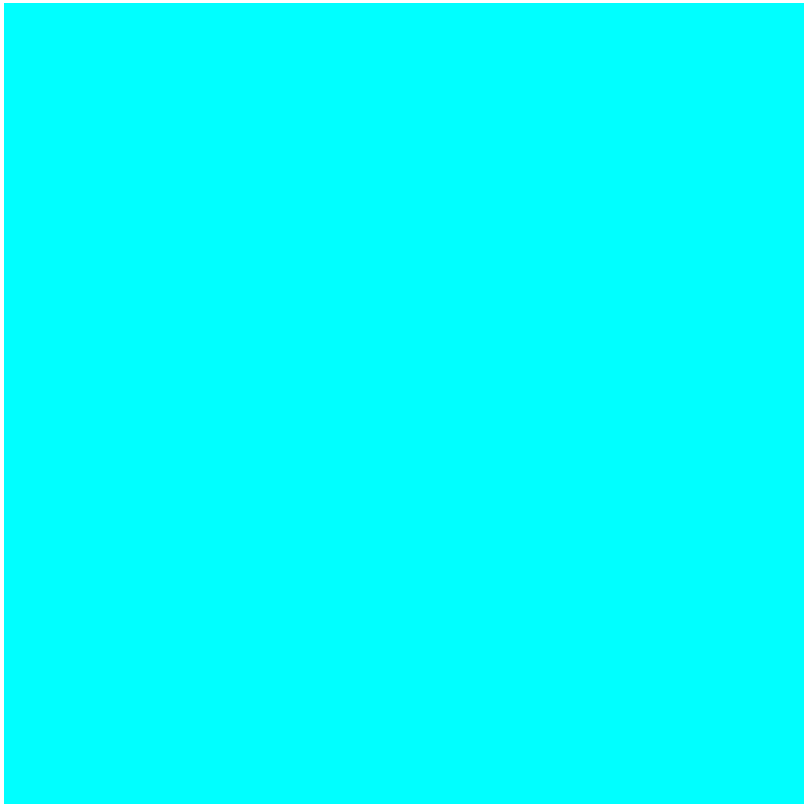
```
function setup() {
  createCanvas(400, 400);
}
function draw() {
  background(220);
}
```

draw() is called directly after setup() and executes the lines of code inside its curly brackets 60 times per second until the program is stopped or the noLoop() function is called.

Can we change the background color?

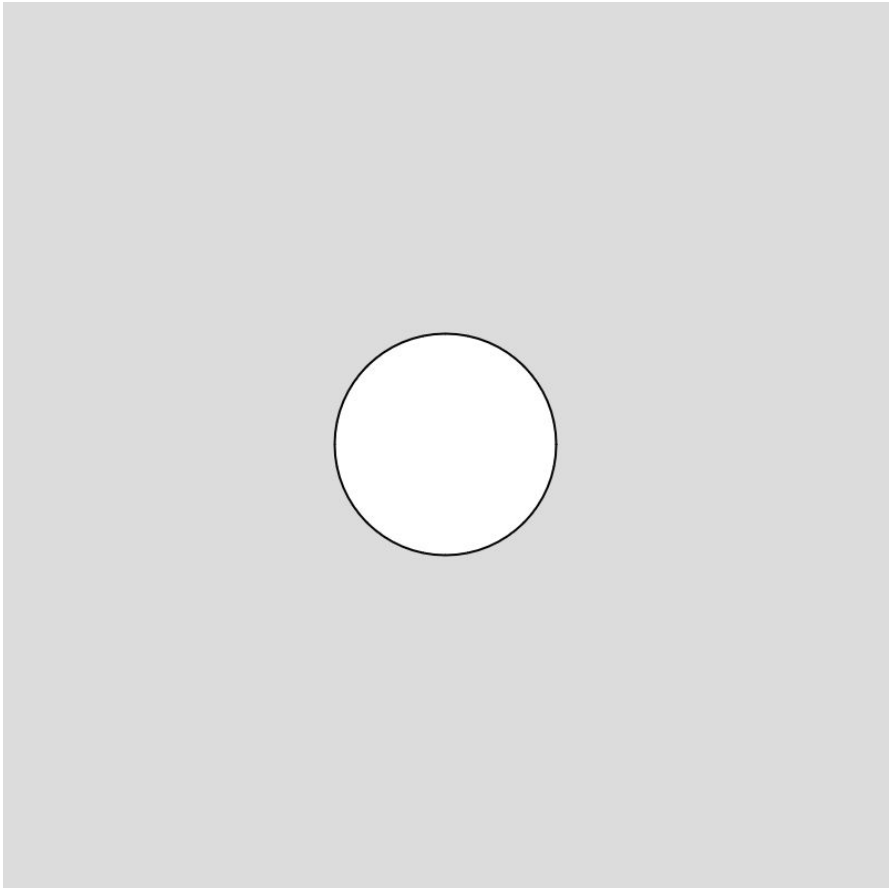


Can we change the background color?



Let's have a look at https://p5js.org/reference/

Can we Draw a Circle with P5.js?



P5 Live Editor

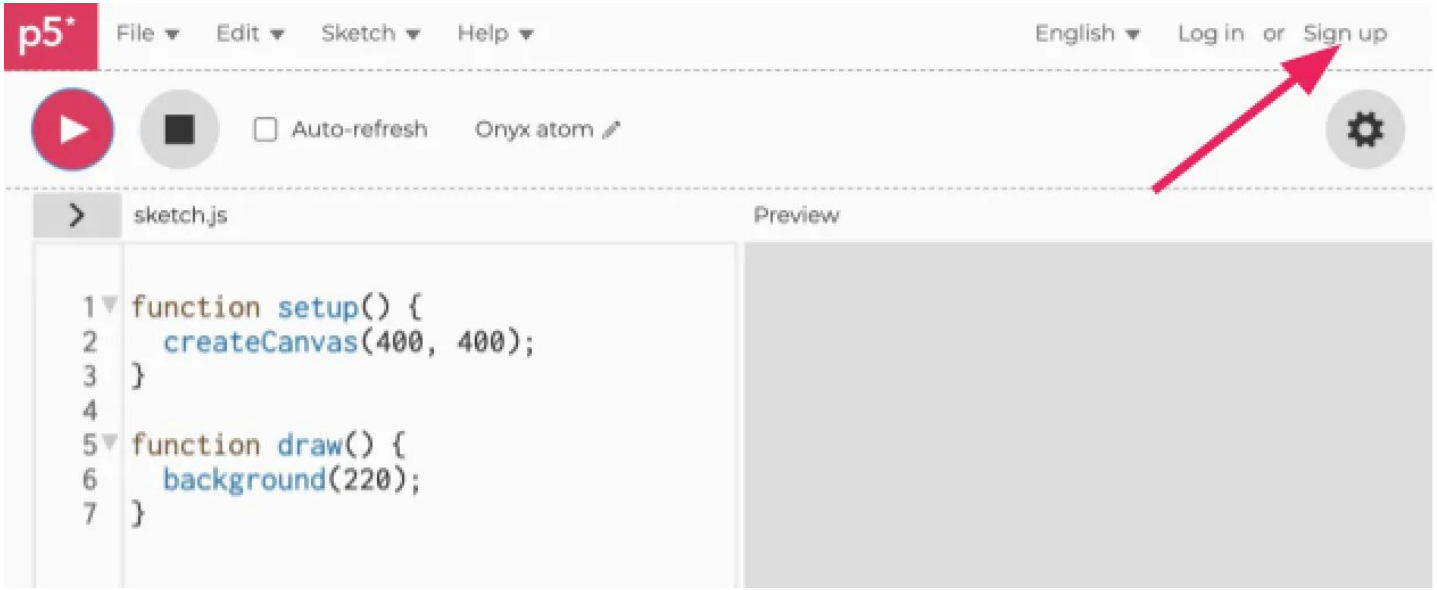
https://editor.p5js.org/

The p5.js Web Editor is a website where programmers can write, test, share, or remix p5.js programs without needing to download or configure a *code editor* on a computer.

P5 Live Editor

https://editor.p5js.org/

The p5.js Web Editor is a website where programmers can write, test, share, or remix p5.js programs without needing to download or configure a *code editor* on a computer.

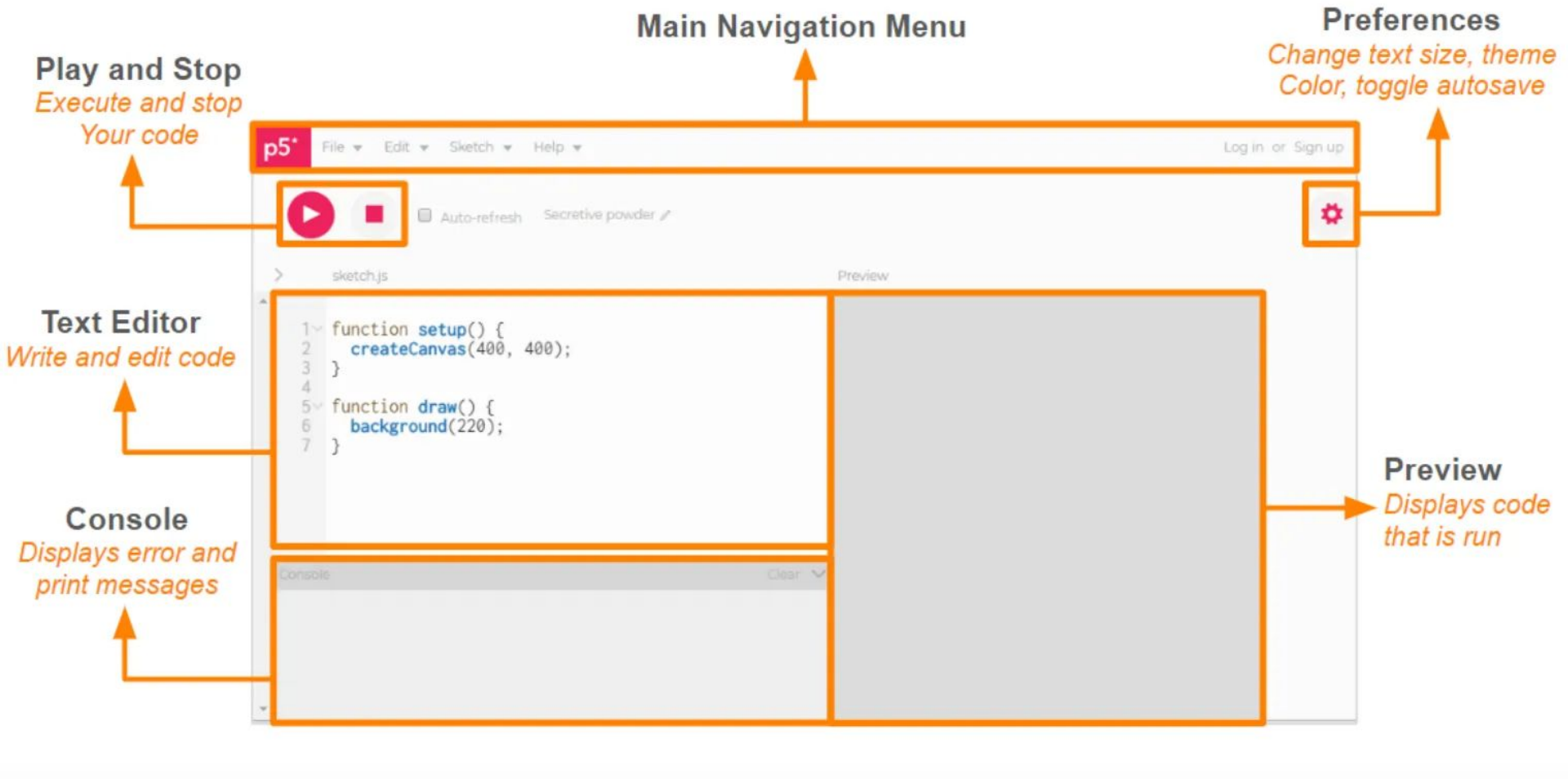


Let’s try to draw a circle with it!

https://editor.p5js.org/

Image by Pietro Merola from Pixabay

p5.js Web Editor Interface



p5.js Drawing a Circle

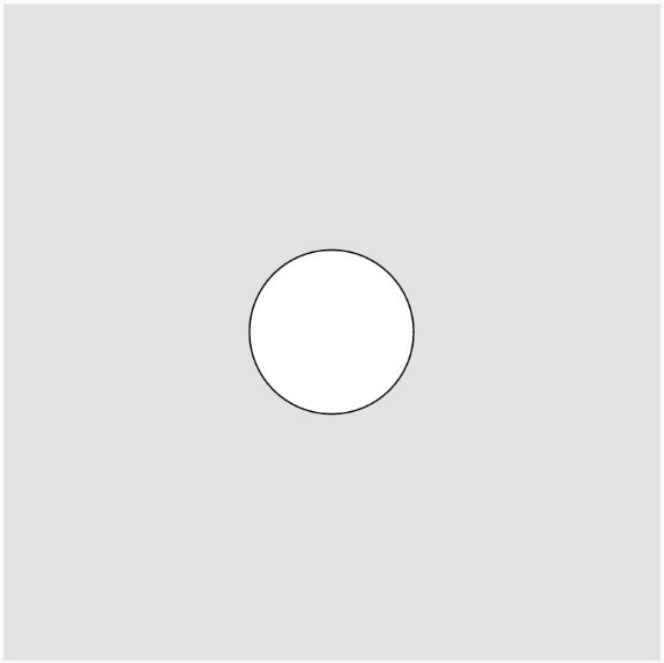
p5* File Edit Sketch Help

Auto-refresh Seed crawdad by DavideConficconi

sketch.js Saved: 17 minutes ago

```
1  function setup() {
2    createCanvas(400, 400);
3  }
4
5  function draw() {
6    background(220);
7    //A circle is a round shape defined by the x, y, and d parameter
8    circle(200,200,100);
9  }
```

Preview



Console Clear

What else can we do with P5.js?

https://p5js.org/tutorials/setting-up-your-environment/

What else can we do with P5.js?

p5.js has many functions you can use to incorporate both ***static*** and ***interactive*** elements in your canvas.

```
//when mouse button is pressed, circles turn black
if (mouseIsPressed === true) {
  fill(0);
} else {
  fill(255);
}
//white circles drawn at mouse position
circle(mouseX, mouseY, 100);
```


https://p5js.org/tutorials/setting-up-your-environment/

~~DON'T~~ Try this at Home

```
function setup() {
   createCanvas(400, 400);
   background(255);
 }
  function draw() {
   circle(mouseX, mouseY, 80);
 }
```


https://p5js.org/tutorials/setting-up-your-environment/

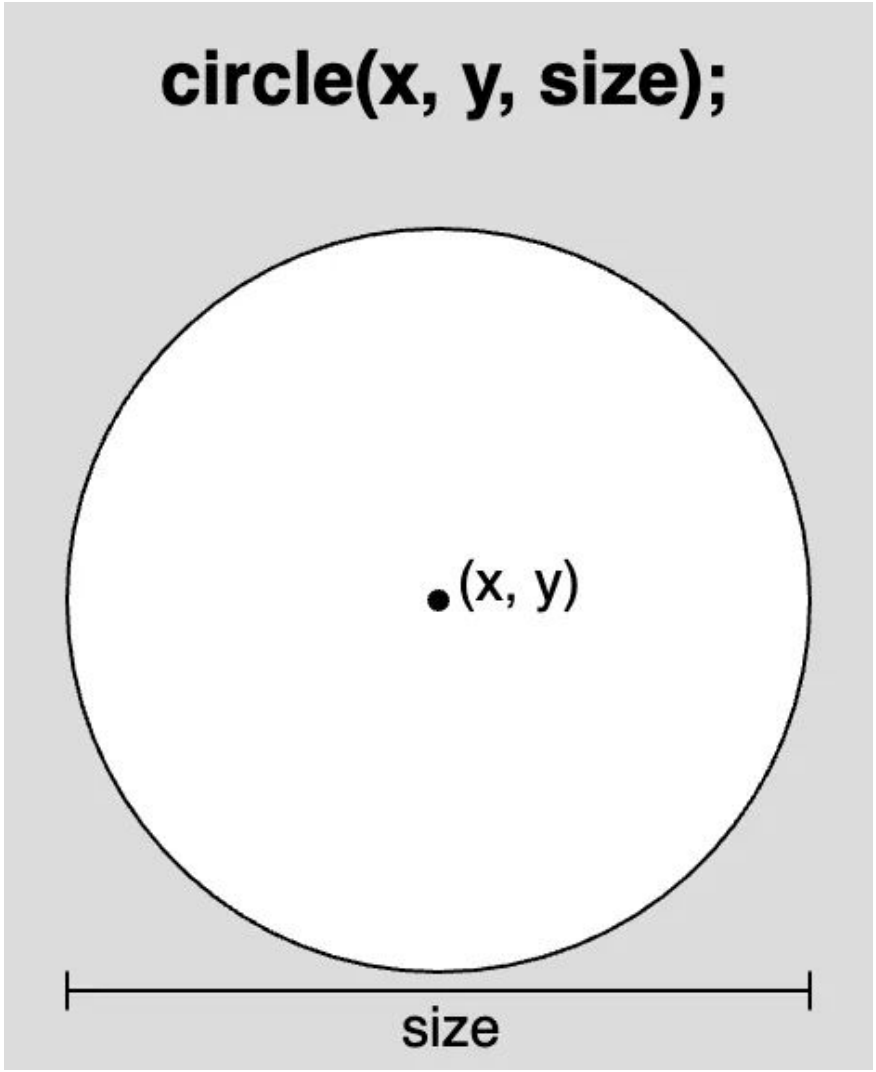
Today Objectives

1. ~~Recap + Challenge discussion~~
2. ~~Refresh on Algorithms~~
3. ~~Dev with P5: Live Editor and Local VScode~~
4. Draw with P5
5. Variables + JS arithmetic
6. Repetitions
7. Branches
8. Put everything live and check github!

https://editor.p5js.org/

Exercise: Draw a Full Moon in the top right corner

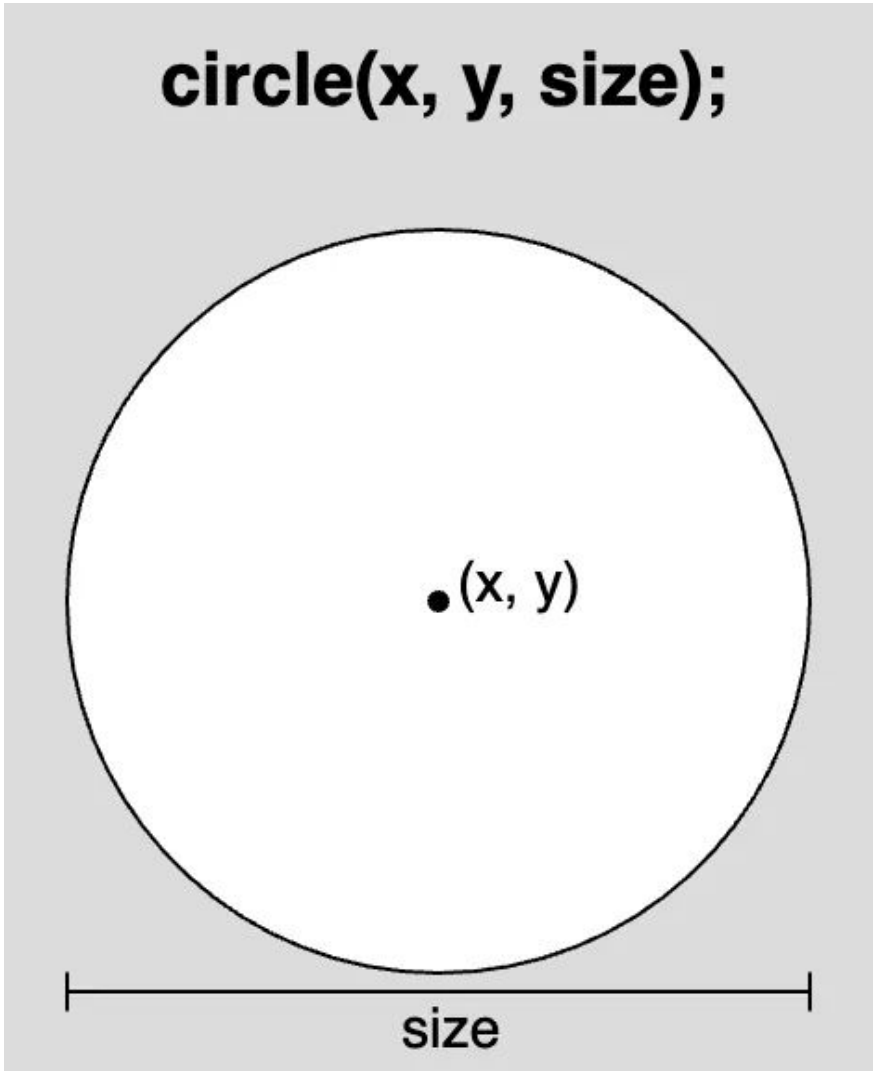
Exercise: Draw a Full Moon in the top right corner

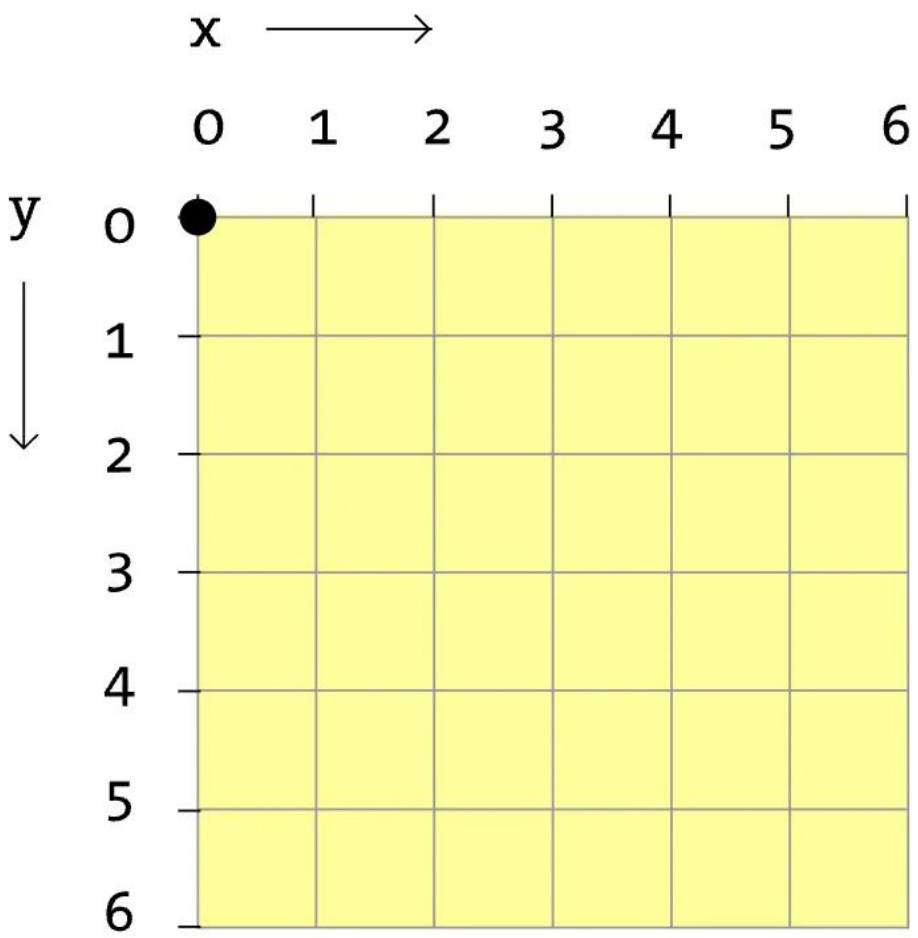


Visit the p5.js reference page for `circle()` to learn more.

https://p5js.org/tutorials/get-started/

Exercise: Draw a Full Moon in the top right corner





Learn more about the HTML canvas coordinate system and shapes on this p5.js reference page.

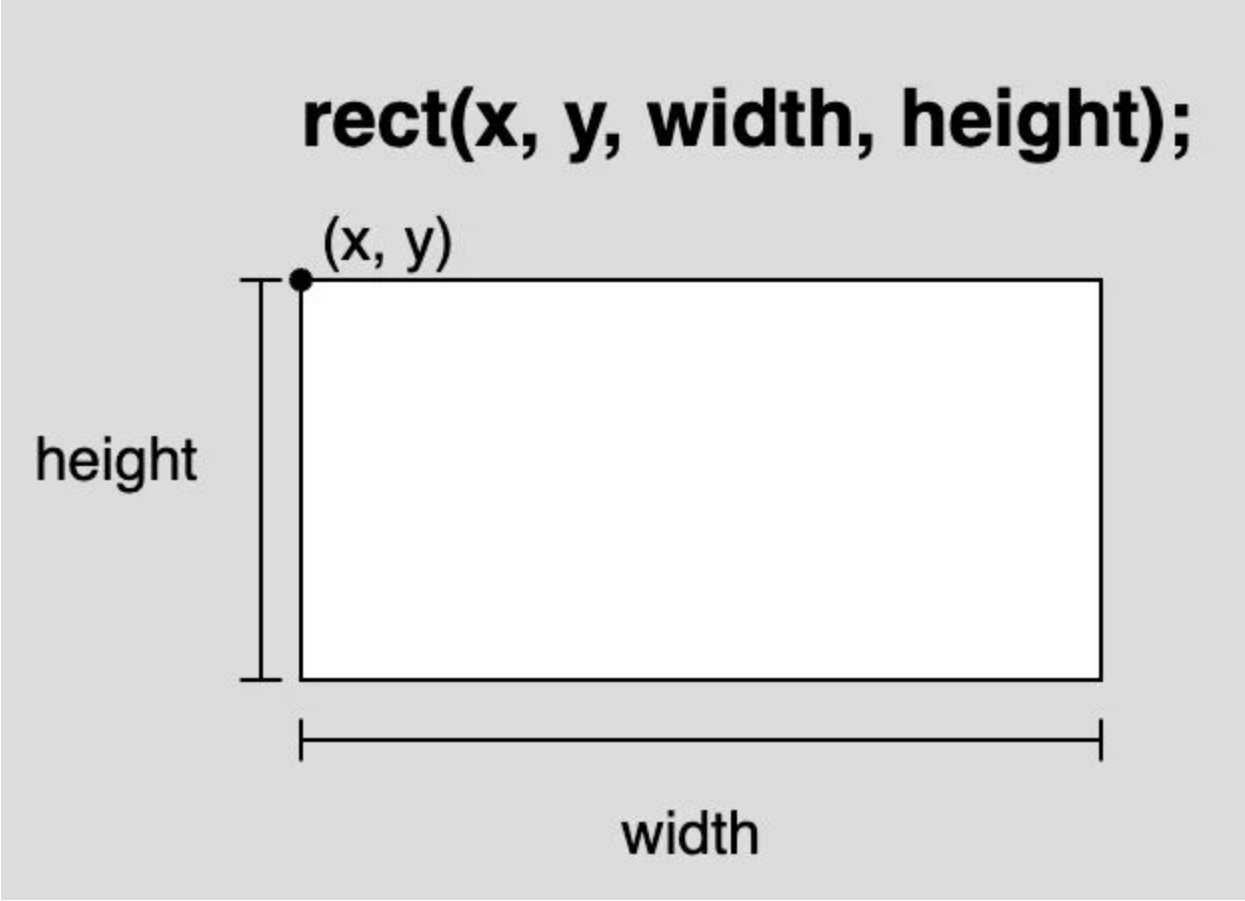
Visit the p5.js reference page for `circle()` to learn more.

https://p5js.org/tutorials/get-started/

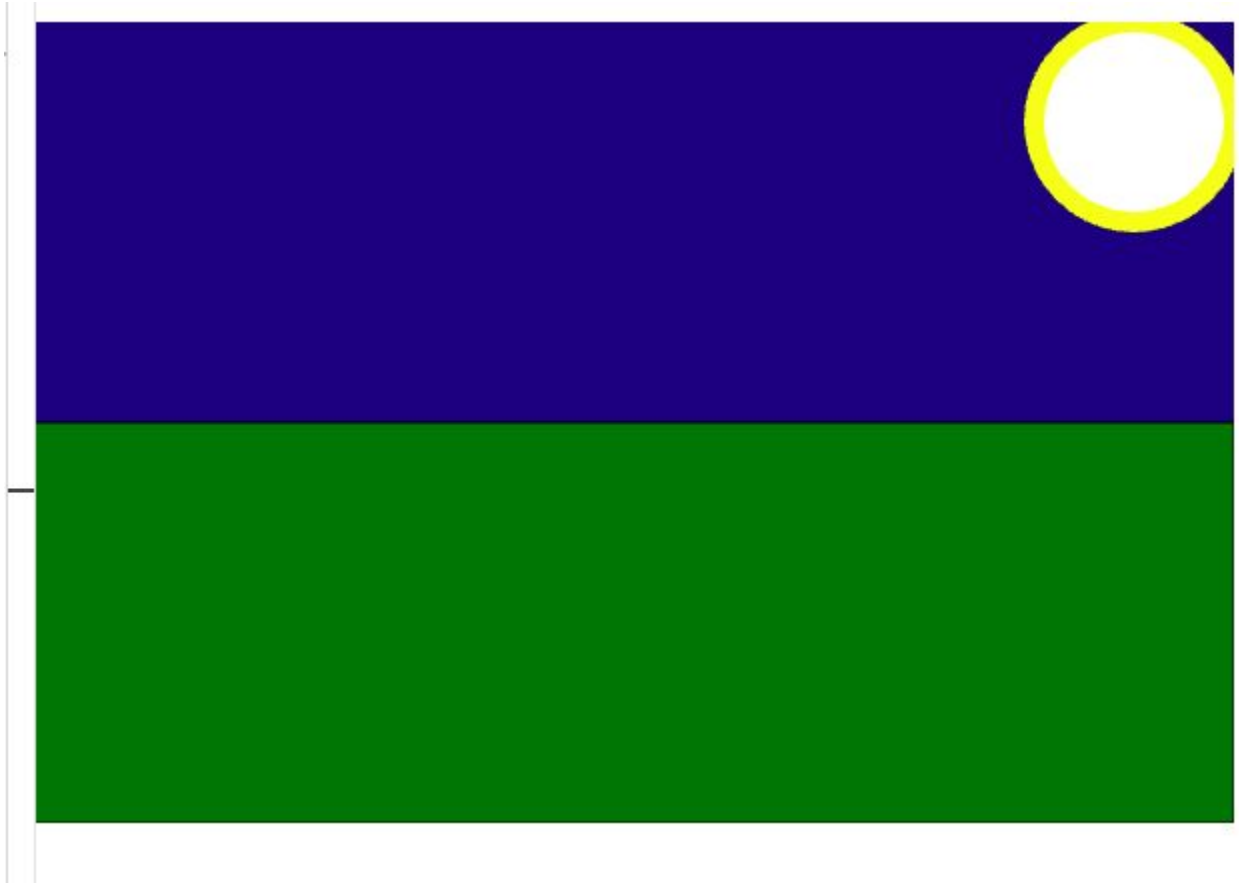
Exercise: Draw a Full Moon in the top right corner

Let’s add the grass

Let’s add the grass



Let’s add the grass



https://p5js.org/tutorials/get-started/

This is only the beginning…

This is only the beginning…

https://p5js.org/tutorials/get-started/

This is only the beginning…

Shape

2D Primitives

`arc()`
Draws an arc.

`circle()`
Draws a circle.

`ellipse()`
Draws an ellipse (oval).

`line()`
Draws a straight line between two points.

`point()`
Draws a single point in space.

`quad()`
Draws a quadrilateral (four-sided shape).

`rect()`
Draws a rectangle.

`square()`
Draws a square.

`triangle()`
Draws a triangle.

https://p5js.org/reference/#Shape

JavaScript (JS) Key 1/2

JavaScript is **interpreted**

– Programs run on top of an **engine without directly translation** to **machine code**

– Advantages: **flexibility**, **ease** of **modification**, …

– Disadvantages: **speed** and (**run**-time) **debugging**

– Other examples: Python, Ruby,...

JavaScript: Datatypes

L'insieme dei tipi di dato in Javascript si dividono in **primitivi** e **oggetti** (vedremo quest'ultimi più avanti)

I tipi primitivi principali sono:

Nome	**Descrizione**	**Esempio**
Boolean	Due possibili stati: vero o falso	*true*
Number	Interi e numeri con la virgola	3.14
String	Sequenze di caratteri	"Informatica applicata"
Null	Un tipo speciale che indica l'assenza di un valore	*null*
Undefined	Un tipo speciale che indica l'assenza di una variable	

JavaScript (JS) Key 2/2

- JavaScript is **dynamically typed**

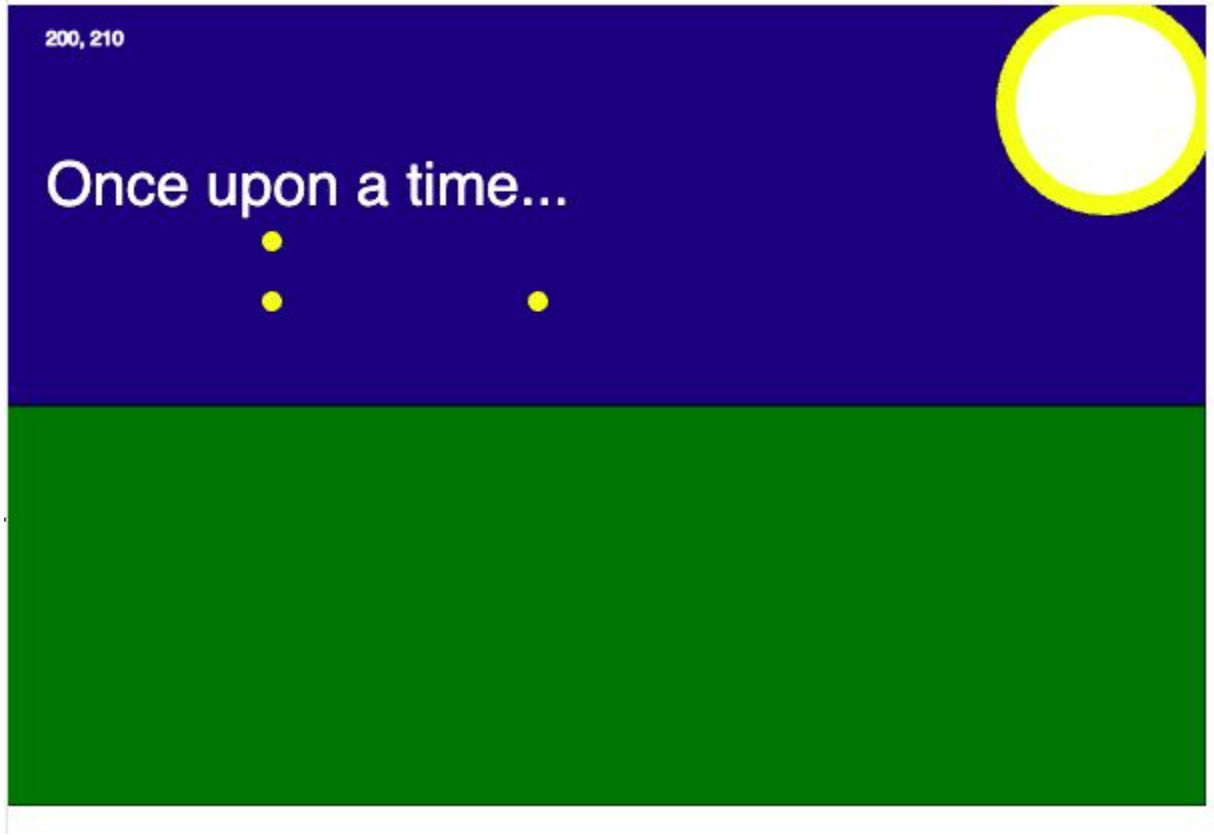
– Variable declaration does **not** define the type

– Types are inferred by the interpreter based on value

– Variables may **change their type** as the program executes

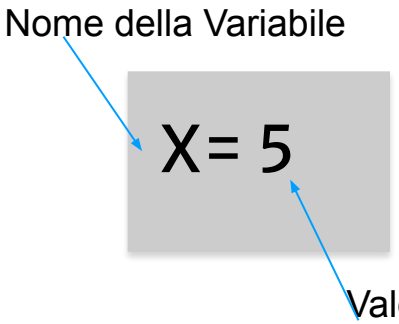
– Type checking **can only happen at run-time**, leaving bugs undiscovered

Let’s add the stars to this sky



https://p5js.org/tutorials/get-started/

Recall: Concetto di Variabile



La variabile X contiene il valore 5

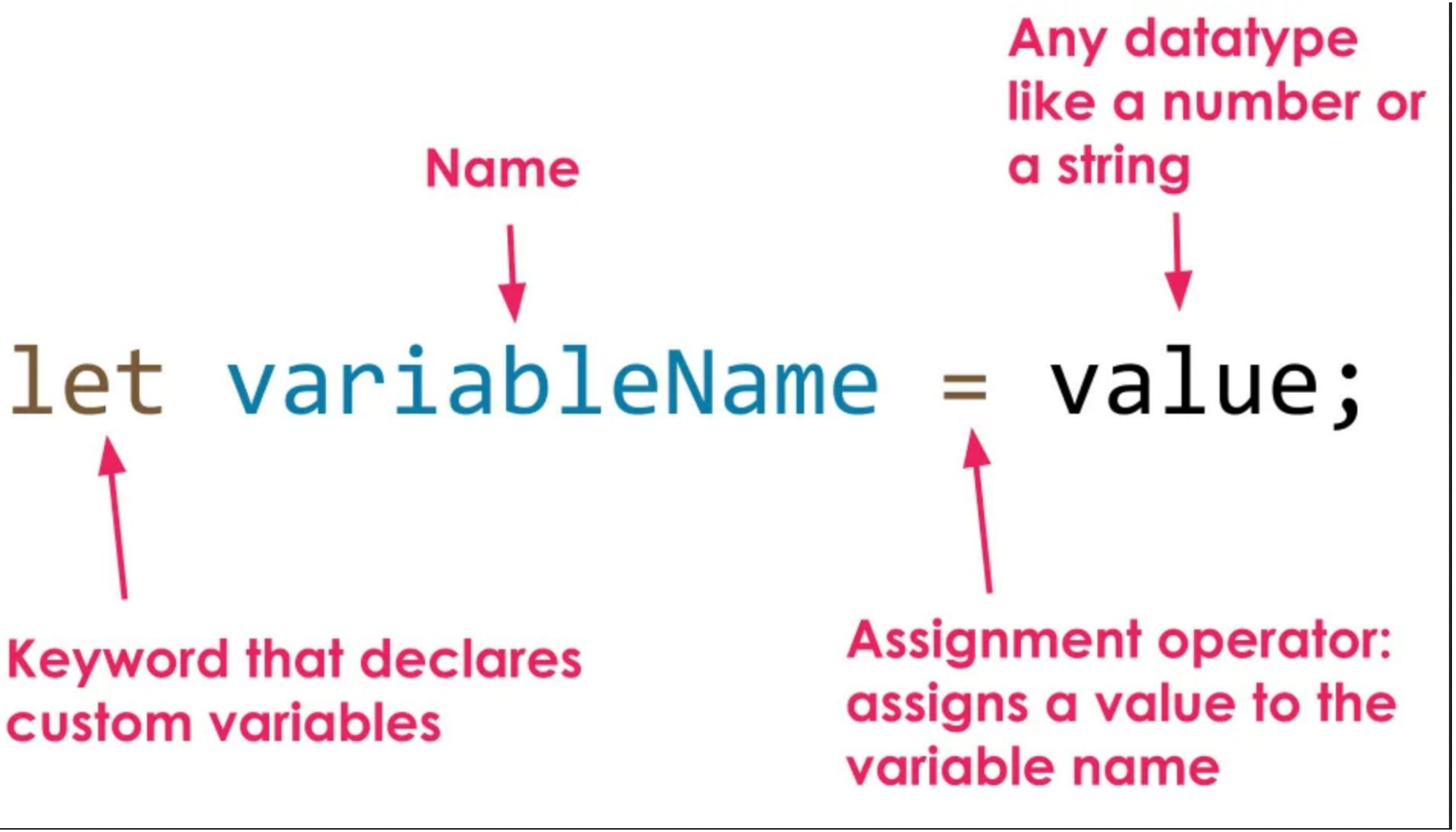
La variabile è un "contenitore" di valori
§ Ad una variabile è sempre associato un valore alla volta
§ La variabile non può essere vuota (al massimo non è inizializzata e quindi contiene un valore "a caso")

- La variabile è "collocata" in una riga della memoria

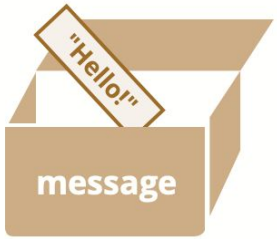
- La variabile è identificata tramite un nome simbolico
§ Le operazioni descritte nell'algoritmo possono essere eseguite di volta in volta sui diversi valori assegnati alle variabili (formulazione generale dell'algoritmo)

- Possiamo eseguire due operazioni su una variabile:
§ Lettura: recuperiamo il valore contenuto
§ Scrittura: salviamo un nuovo valore

JS Variables

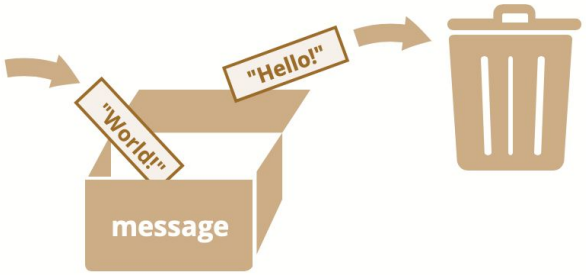


JS Variables and Constants



Variables are named storage for data
– Like in any other language…
– They are initialized, read and written
– Declaring twice triggers an error
– Case matters, symbol $ and _ are legal

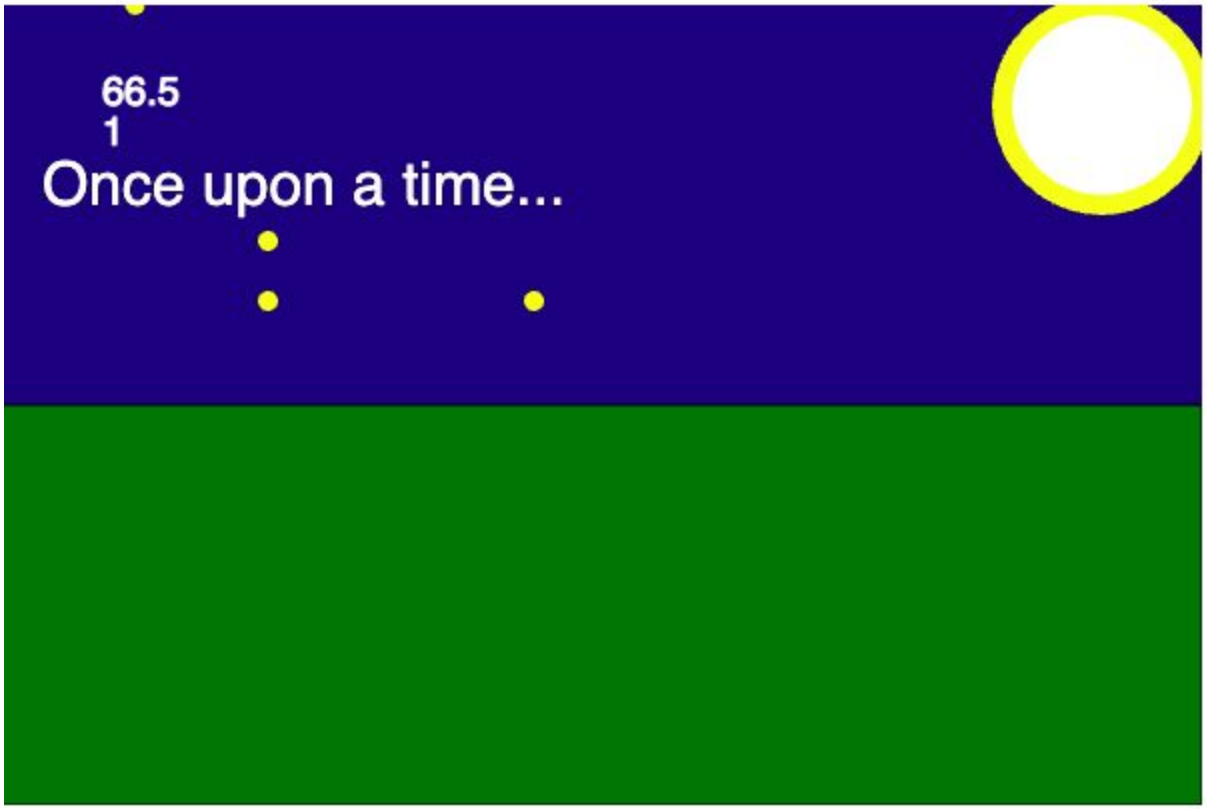
```
let message = ”Hello!”;
console.log(message);
```



Constants use `const` instead of `let`

```
let message = ”Hello!”;
message = ”World!”;
console.log(message);
```

If we use variables



JS: Operatori Matematici

Binari

Simbolo	Descrizione
+	Somma
-	Differenza
*	Moltiplicazione
/	Divisione
%	Resto della divisione intera
**	Elevamento a potenza

Unari

Simbolo	Descrizione
++a	Pre-Incremento
a++	Post-Incremento
--a	Pre-Decremento
a--	Post-Decremento

Gli operatori binari possono anche essere utilizzati come assegnamento:

`a = a + 5` può essere scritto come `a += 5`

L'operatore di **divisione /** calcola:

`13 / 5` è uguale `2.6`

L'operatore **Modulo** calcola il **resto** della divisione:

`13 % 5` è uguale `3`

JS: Operatori di Confronto

Simbolo	Descrizione
==	Uguale
!=	Diverso
===	Strettamente Uguale *
!==	Strettamente Diverso *
> e >=	Maggiore e Maggiore Uguale
< e <=	Minore e Minore Uguale

* L'operatore **strettamente** uguale verifica che anche il **tipo di dato** sia equivalente, senza eseguire il casting.

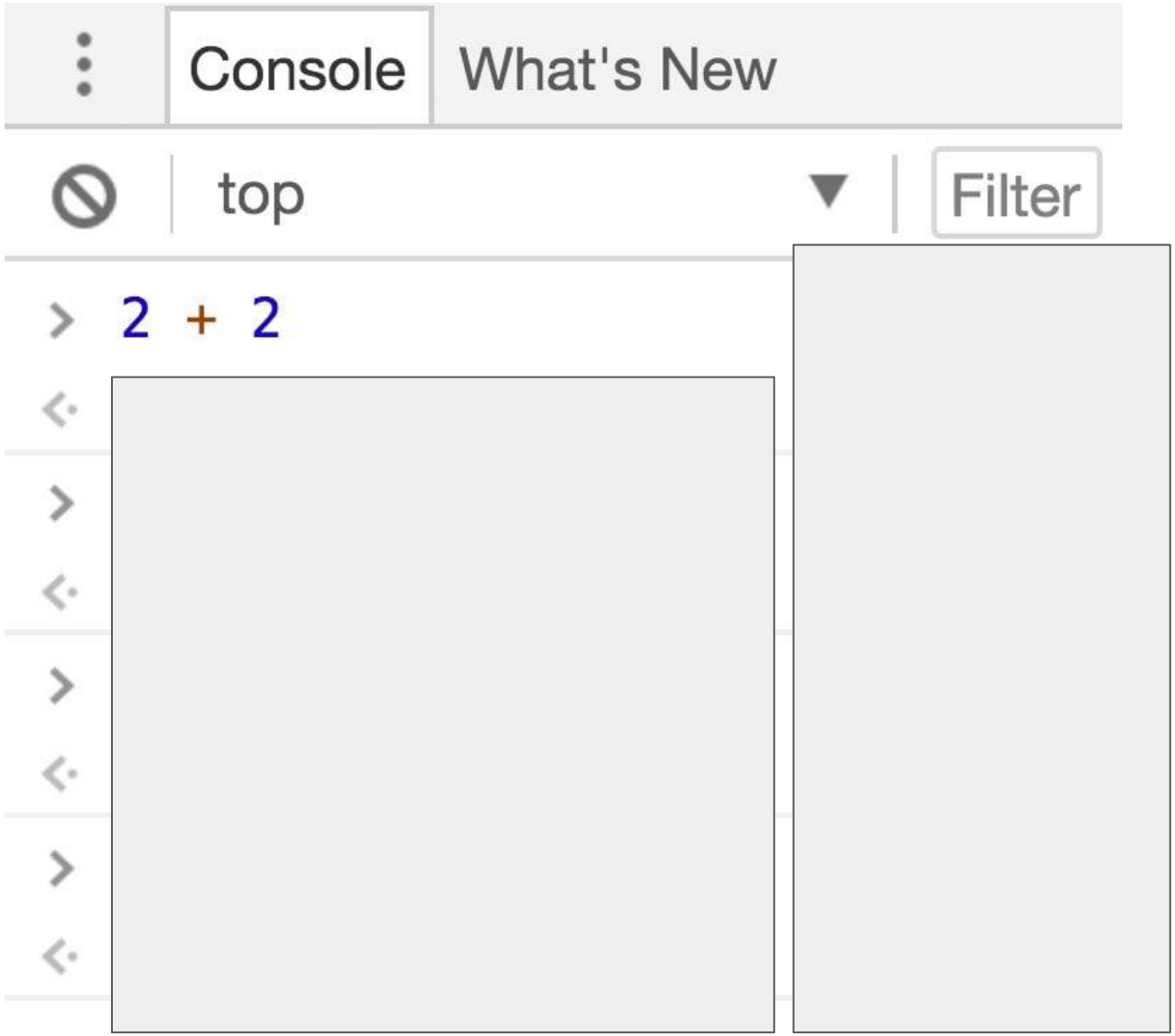
In modo analogo funziona lo strettamente diverso.

Esempio:

- 5 == 5 e 5 === 5 sono veri
- 5 == "5" è vero
- 5 != "5" è falso
- 5 === "5" è falso
- 5 !== "5" è vero

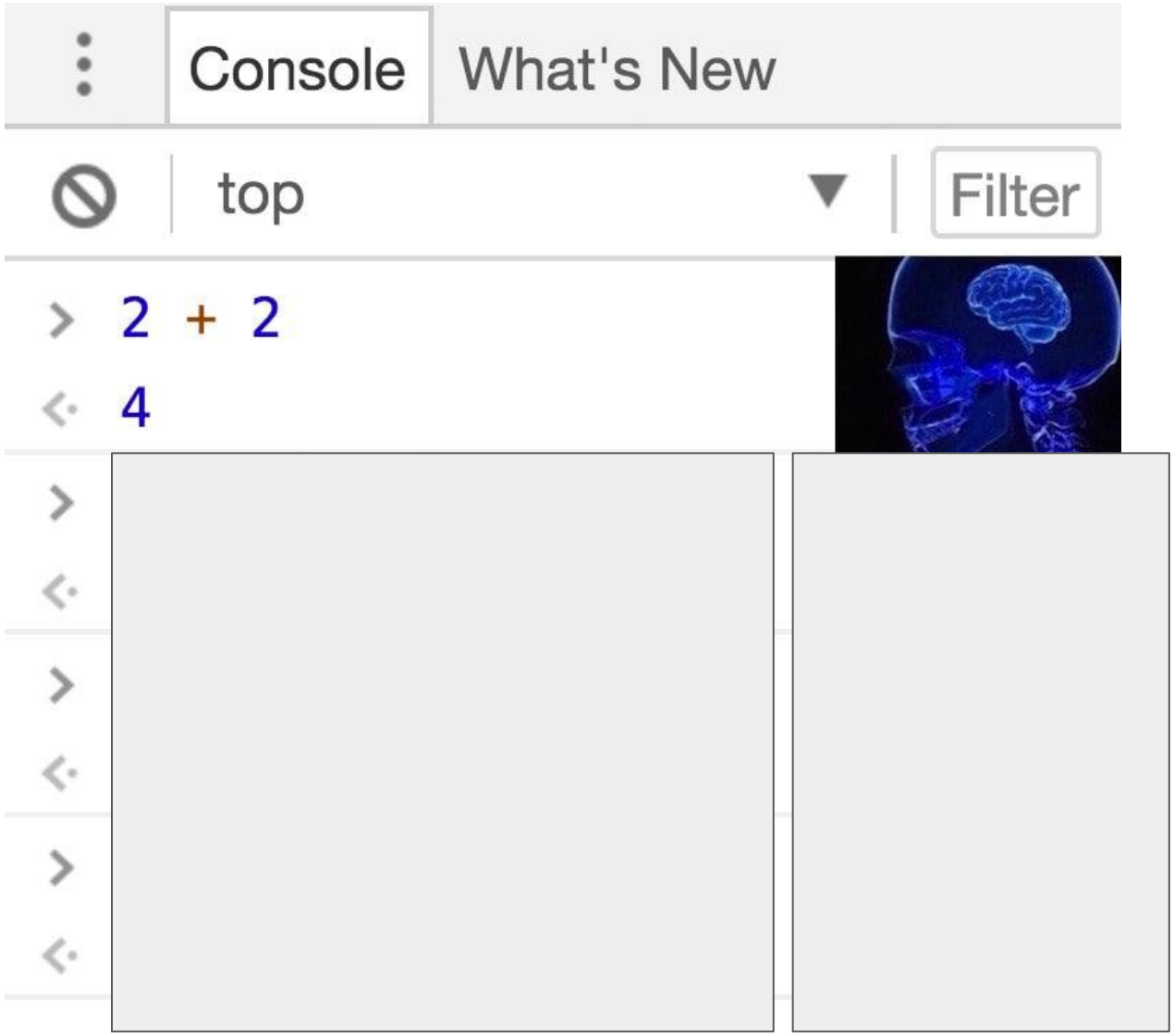
When comparing **different types**, everything is **converted** to **Number**
– This leads to funny consequences… make sure what type you are comparing with what else!
– The value undefined cannot be compared to anything, yields false

Let's Give a Try



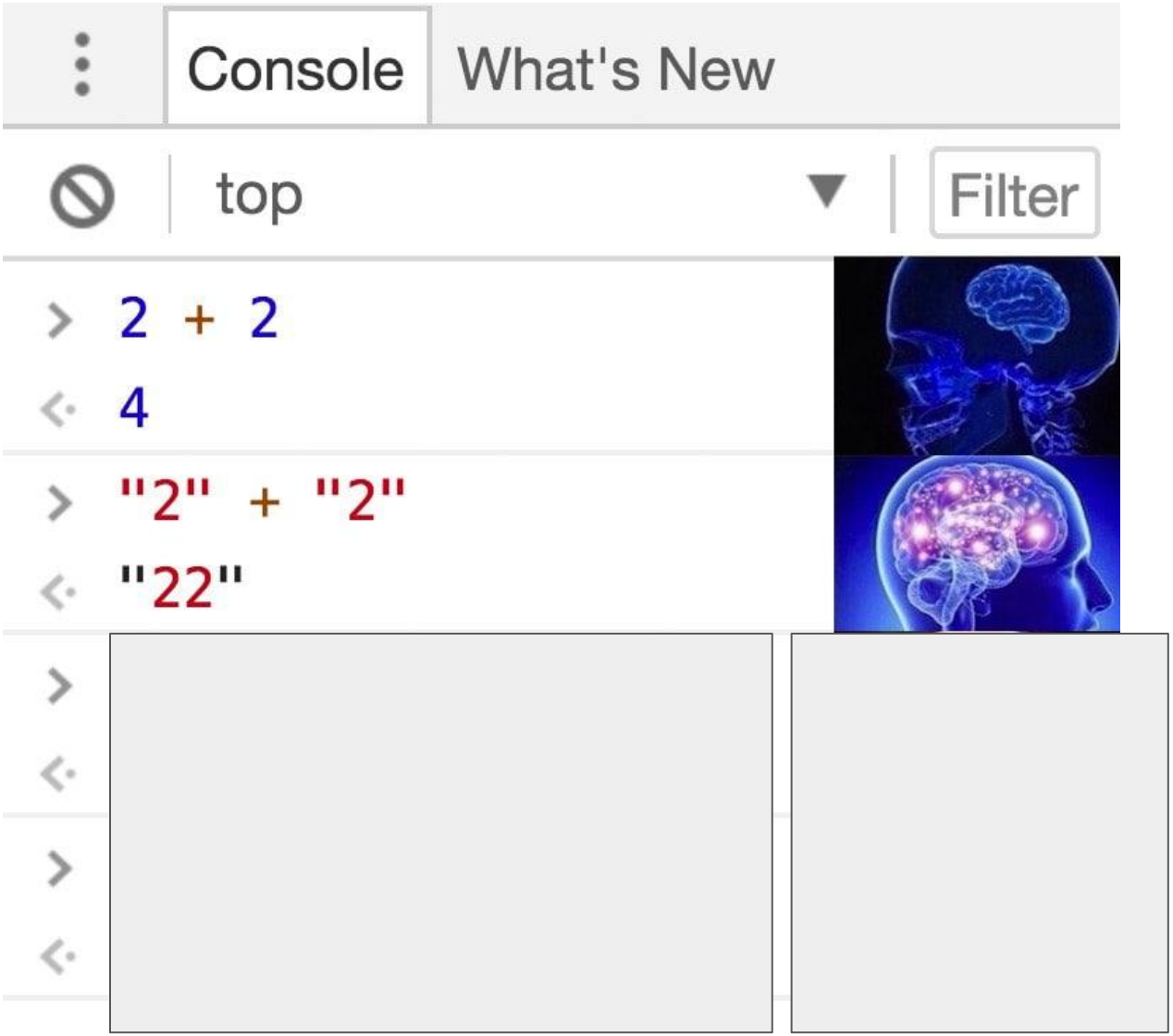
https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

Let's Give a Try



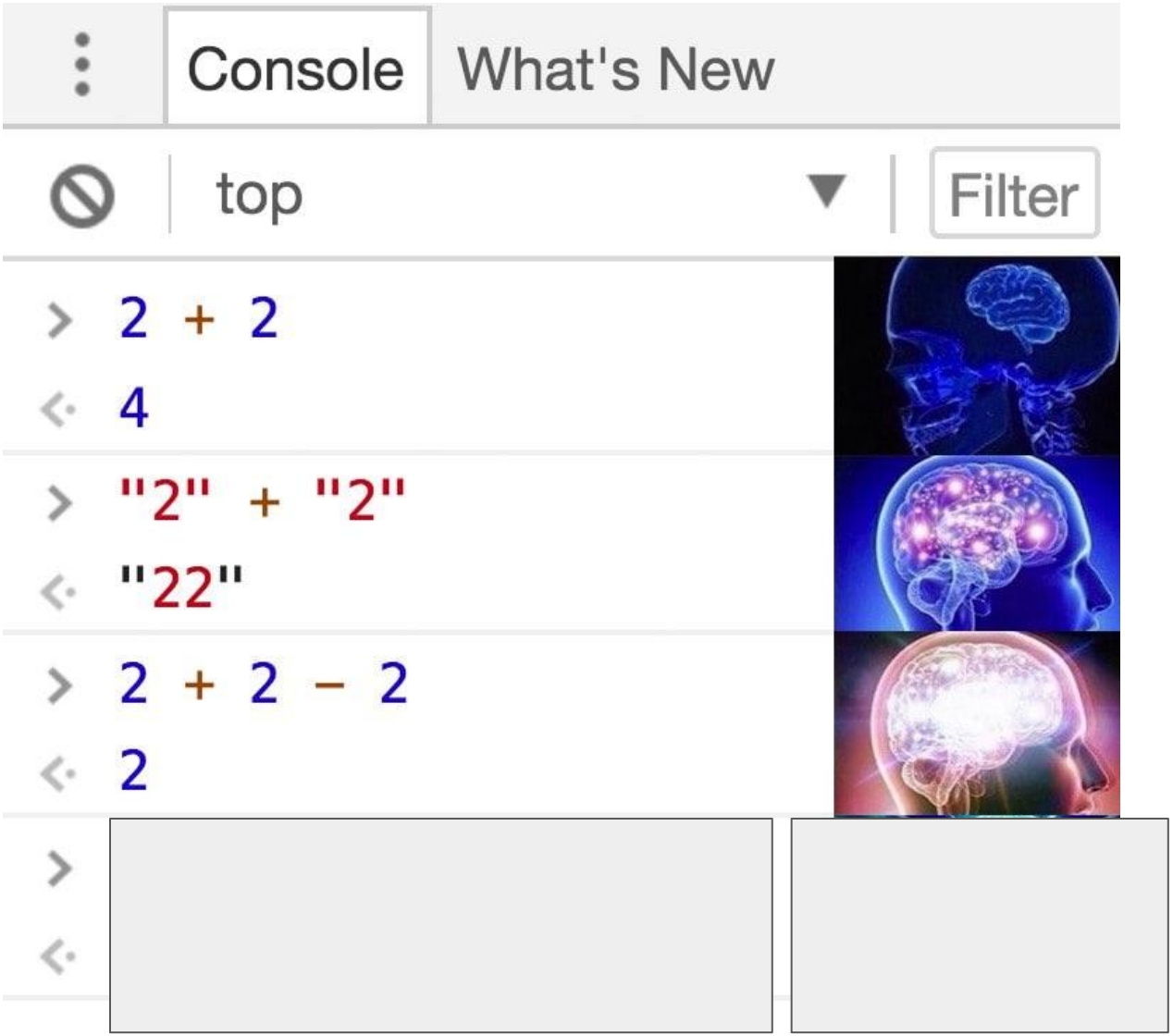
https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

Let's Give a Try



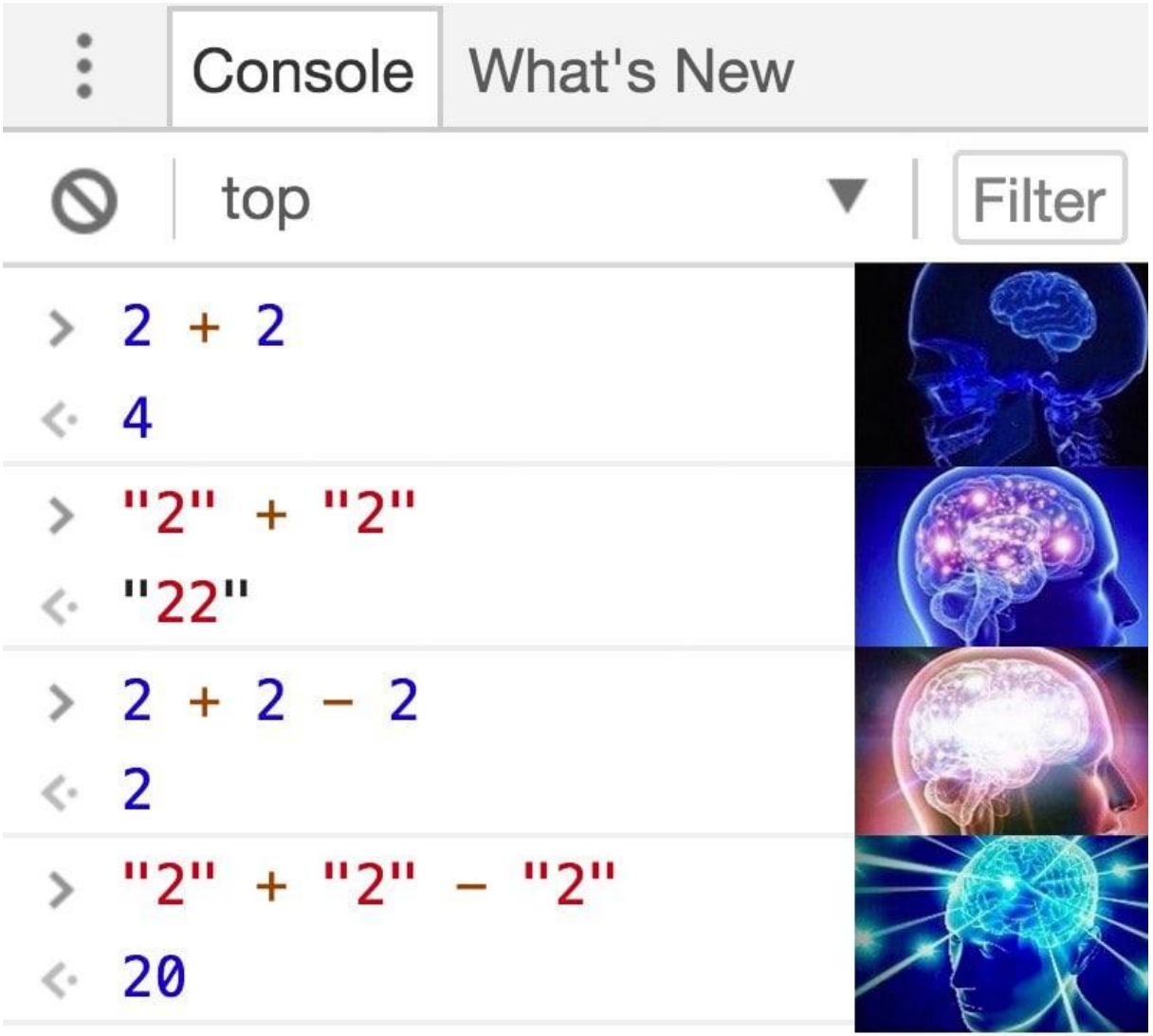
https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

Let's Give a Try



https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

Let's Give a Try



https://dev.to/damxipo/javascript-versus-memes-explaining-various-funny-memes-2o8c

JS: Operatori Logici

Simbolo	Nome	Descrizione
&&	and	Entrambi gli operandi devono essere veri
||	or	Almeno uno degli operandi deve essere vero
!	not	Inverte il valore dell'operando

A	B	A or B
0	0	0
0	1	1
1	0	1
1	1	1

(somma logica)

A	B	A and B
0	0	0
0	1	0
1	0	0
1	1	1

(prodotto logico)

A	not A
0	1
1	0

(negazione)

Precedenza: l'operatore "not" precede l'operatore "and", che a sua volta precede l'operatore "or"

A and not B or B and C = (A and (not B)) or (B and C)

Tabella di verità di un'espressione logica

A and B or not C

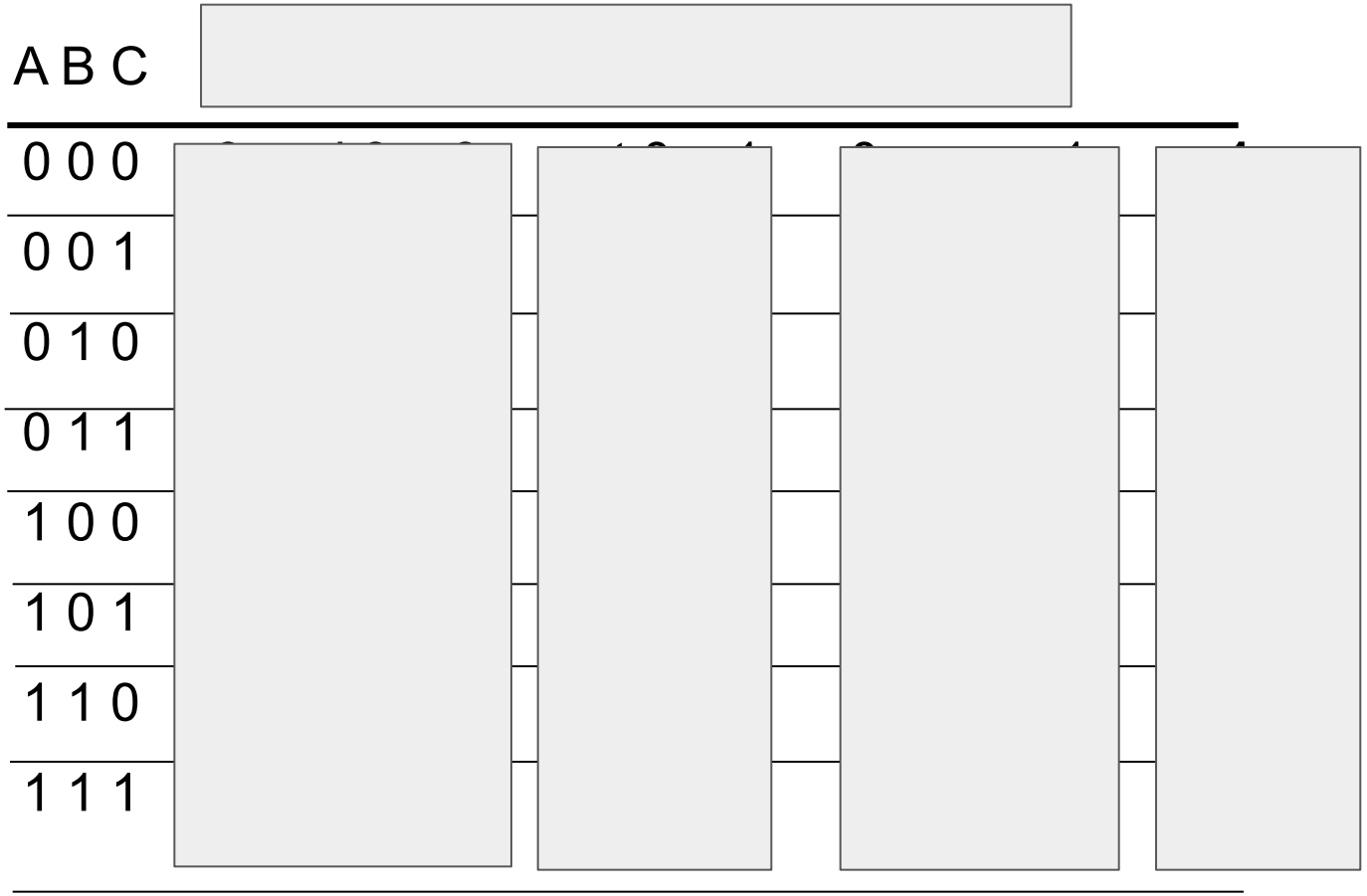


Tabella di verità di un'espressione logica

A and B or not C

A B C	**X** = A and B	**Y** = not C	**X** or **Y**
0 0 0	0 and 0 = 0	not 0 = 1	0 or 1 **= 1**
0 0 1	0 and 0 = 0	not 1 = 0	0 or 0 **= 0**
0 1 0	0 and 1 = 0	not 0 = 1	0 or 1 **= 1**
0 1 1	0 and 1 = 0	not 1 = 0	0 or 0 **= 0**
1 0 0	1 and 0 = 0	not 0 = 1	0 or 1 **= 1**
1 0 1	1 and 0 = 0	not 1 = 0	0 or 0 **= 0**
1 1 0	1 and 1 = 1	not 0 = 1	1 or 1 **= 1**
1 1 1	1 and 1 = 1	not 1 = 0	1 or 0 **= 1**

JS Variables Scope

```
let totale = 0;

function compute_area(base, height) {
    let area;
    area = base * height;
    totale += area;
    return area;
}

compute_area(5,5);
console.log(totale);
```

Local variable that exists only within the function scope

Functions can access also **global** variables

JS Variables Scope

```
let totale = 0;

function compute_area(base, height) {
    let area;
    area = base * height;
    totale += area;
    return area;
}

compute_area(5,5);
console.log(totale); // Stampa 25
```

Local variable that exists only within the function scope

Functions can access also **global** variables

JS Variables Scope cont'd

```
let totale = 0;

function compute_area(base, height) {
    let area;
    area = base * height;
    let totale=0;
    totale += area;
    console.log(totale); // Stampa 25
    return area;
}

compute_area(5,5);
console.log(totale); // Stampa 0
```

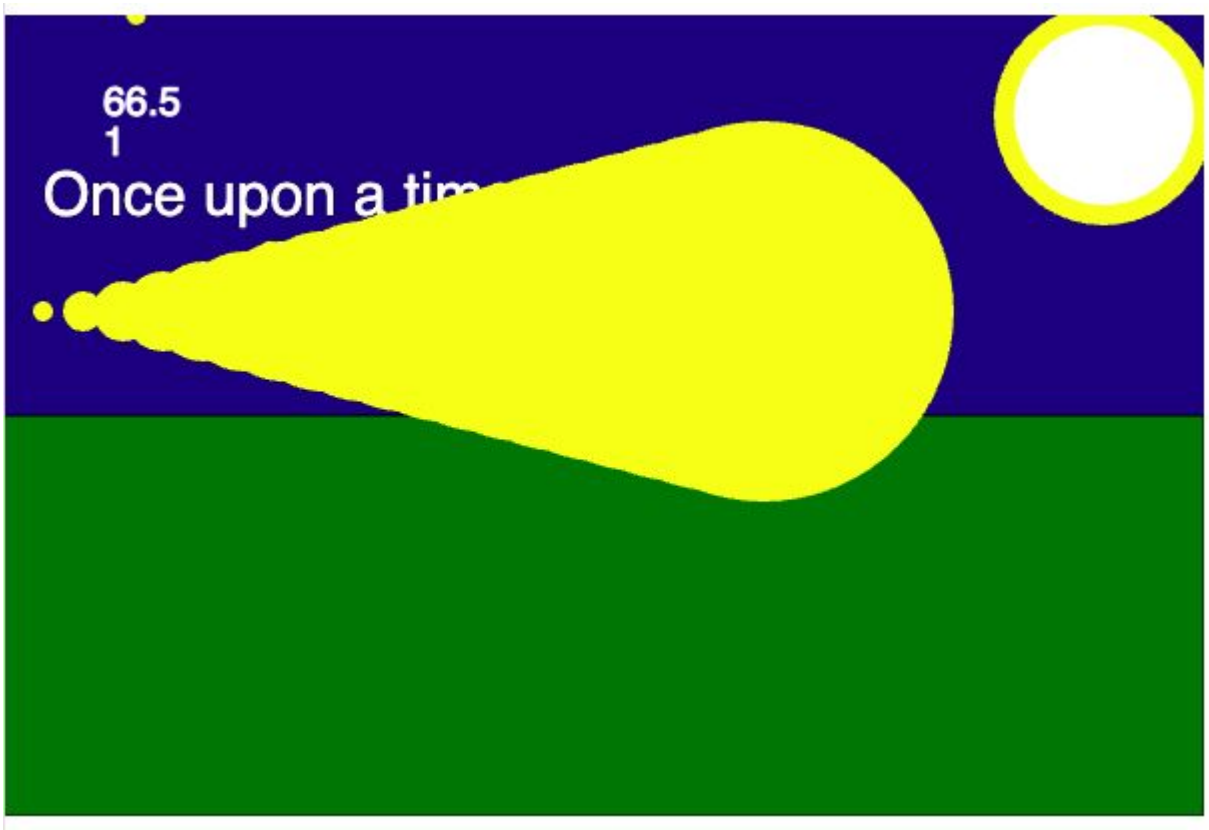
WARNING!!!
If defined a local variable with the same name, then global variable not accessible anymore

Today Objectives

1. ~~Recap + Challenge discussion~~
2. ~~Refresh on Algorithms~~
3. ~~Dev with P5: Live Editor and Local VScode~~
4. ~~Draw with P5~~
5. ~~Variables + JS arithmetic~~
6. Repetitions
7. Branches
8. Put everything live and check github!

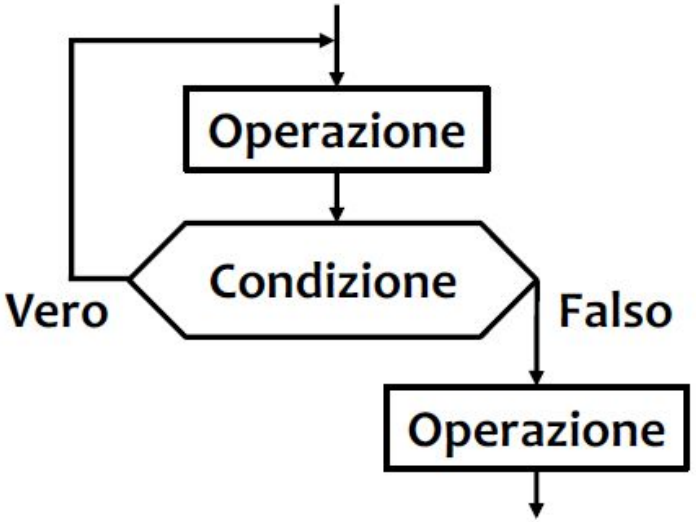
https://editor.p5js.org/

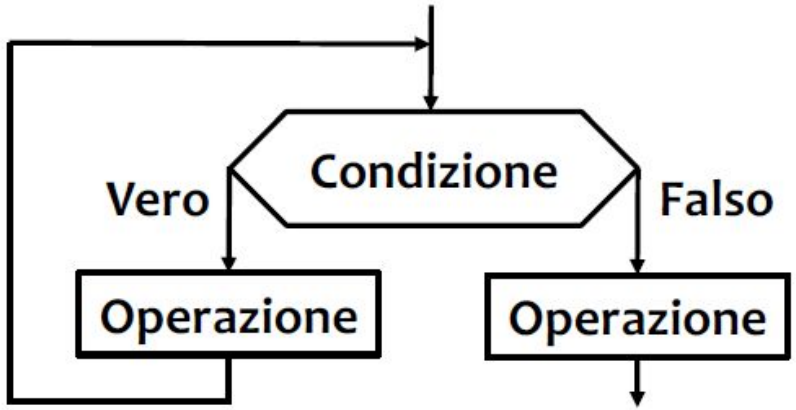
If we want to repeat (a.k.a.) looping



https://p5js.org/tutorials/get-started/

Recall: Iterazione



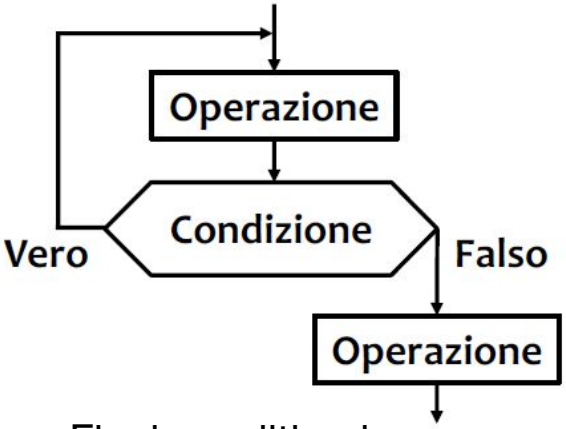


Ciclo a condizione finale:
§ il corpo del ciclo è eseguito almeno una volta

Ciclo a condizione iniziale:
§ il corpo del ciclo è eseguito zero o più volte

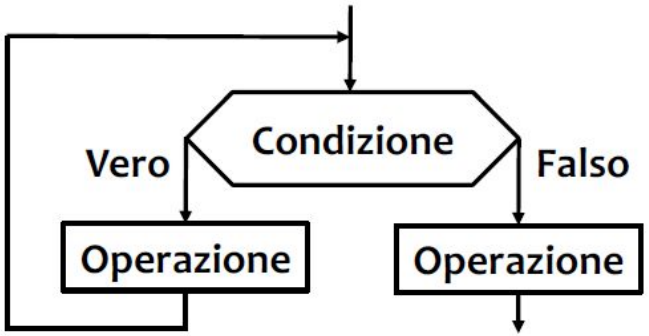
- Corrisponde a un do while in JS
- Corrisponde a un while/for in JS

JS: Loops



Final condition loop:
Body executed at least once

→ **`do ... while`**



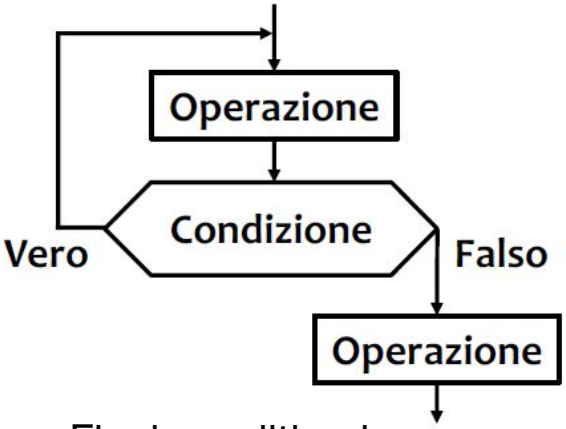
Initial condition loop:
Body executed zero or more

→ **`while, for`**

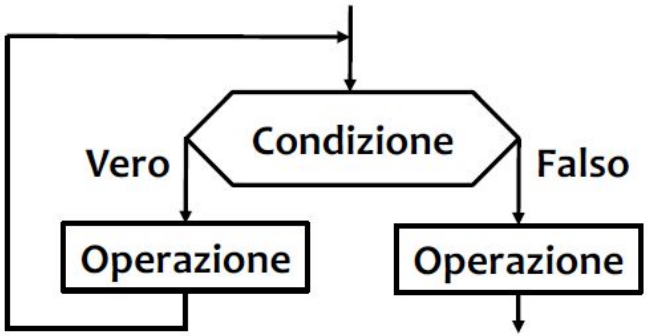
```
for (let i=0; i<10; i++) {
    console.log(i);

}
```

JS: Loops



Final condition loop:
Body executed at least once
→ **do … while**



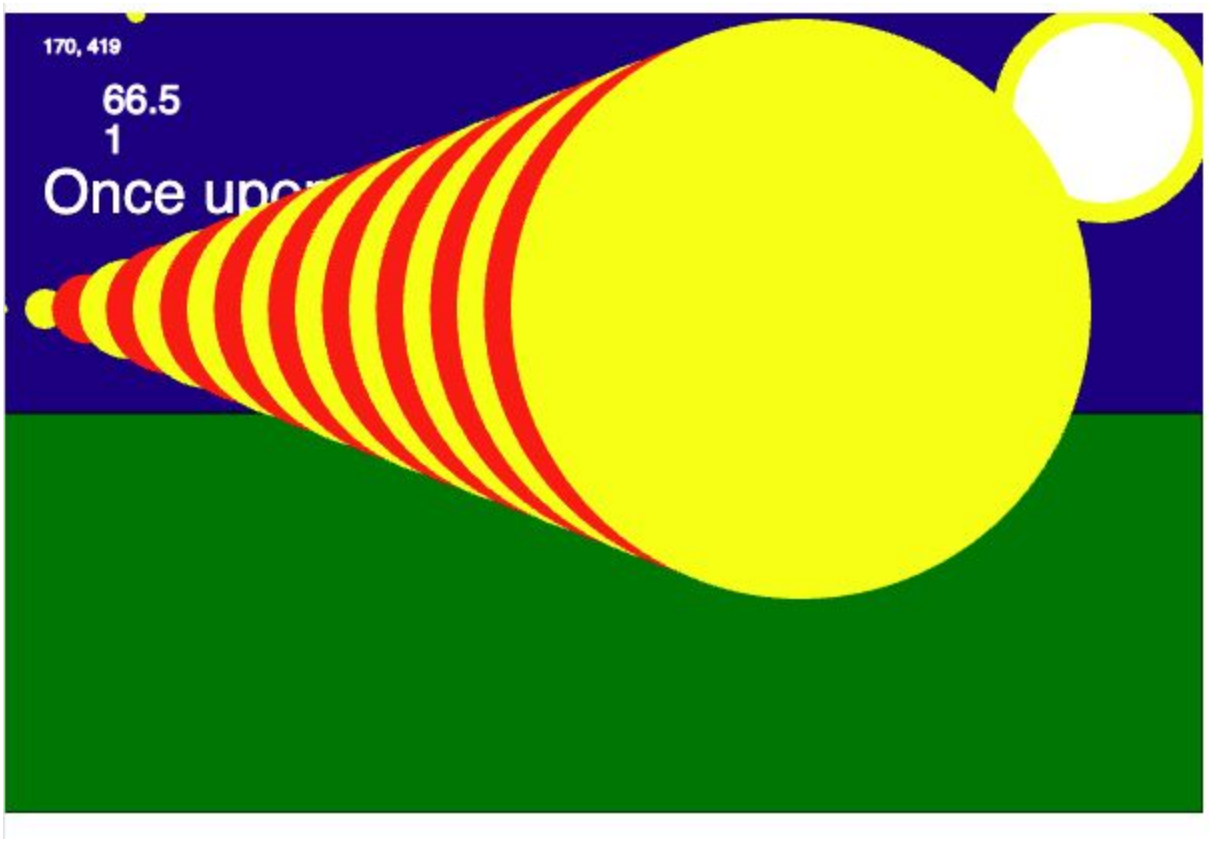
Initial condition loop:
Body executed zero or more
→ **while, for**

Loops may be boken with **break**, or skip the current iteration with **continue**

```
for (let i=0; i<10; i++) {
    console.log(i);
    if (i==8) break;
}
```

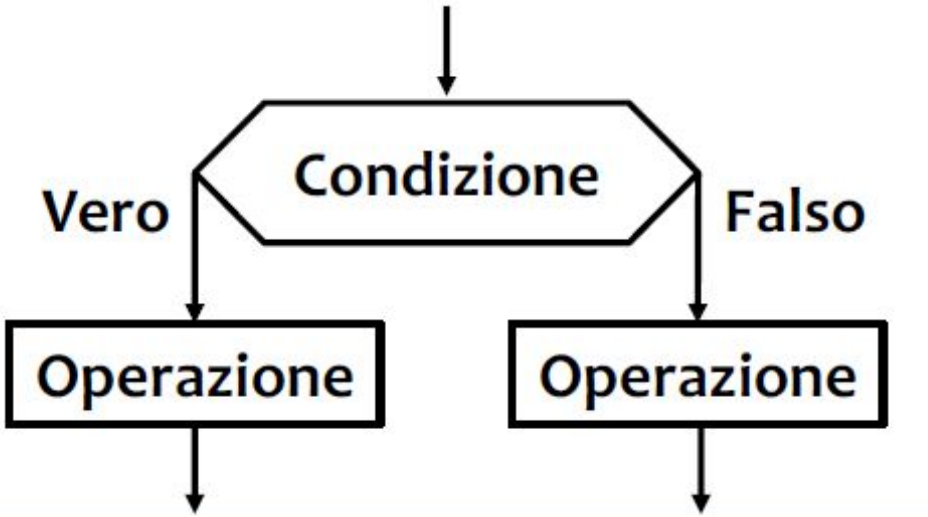
Effectively executes only nine iterations!

If we want to do conditional things (a.k.a. branches)



https://p5js.org/tutorials/get-started/

Recall: La selezione



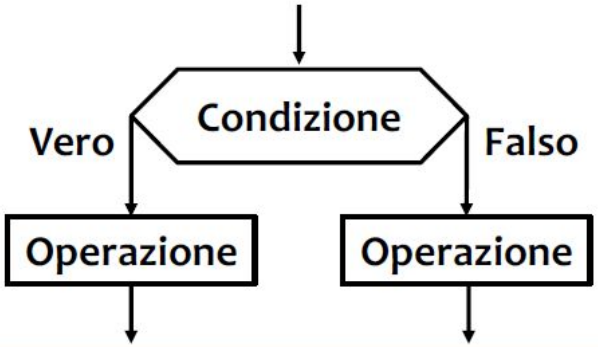
Il blocco condizionale modifica il flusso di controllo rendendolo non più sequenziale

- Dal blocco condizionale escono due frecce con il valore **vero** ed una con il valore **falso** (l'esecuzione dipende dal valore logico risultate dalla valutazione della condizione)
- Durante l'esecuzione, la scelta del blocco da eseguire è sempre **univoca**

 Corrisponde al costrutto If-else in JS

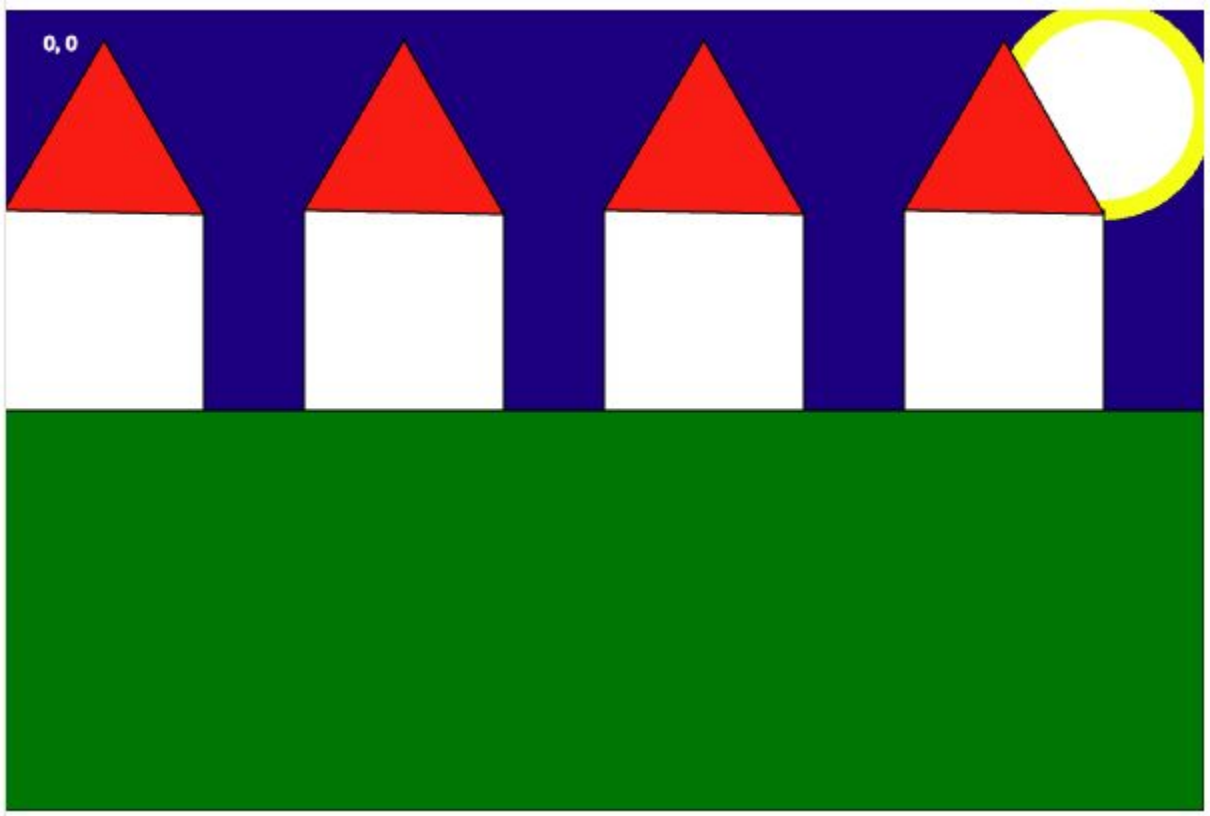
JS: Branching

Branching with if works the usual way
– Logical operators too!
– The and (or) operator evaluates to the first false (true) value, or the last one
• Consider the type conversion rules
– 0, “”, null, undefined, and NaN all become false
– Everything else becomes true
• Also switch is available..




```
if (date.getMonth()==0) {
    console.log("January");
} else if (date.getMonth()==1) {
    console.log("February");
} else {
    console.log("March and later..");
}
```

Exercise: Try to draw some houses



Today Objectives

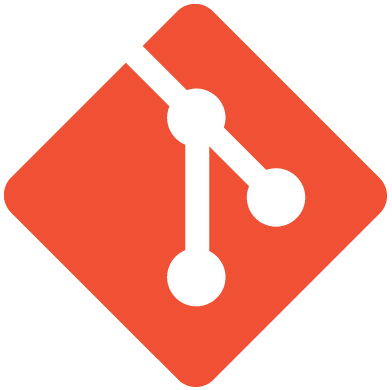
1. ~~Recap + Challenge discussion~~
2. ~~Refresh on Algorithms~~
3. ~~Dev with P5: Live Editor and Local VScode~~
4. ~~Draw with P5~~
5. ~~Variables + JS arithmetic~~
6. ~~Repetitions~~
7. ~~Branches~~
8. Put everything live and check github!

https://editor.p5js.org/

Let’s work with git and put it live

https://pixabay.com/photos/kingfisher-bird-animal-feathers-9016787/

Recall:

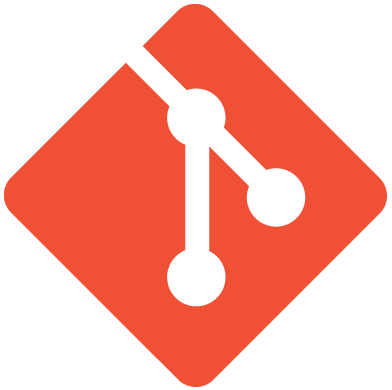


git

Recall:

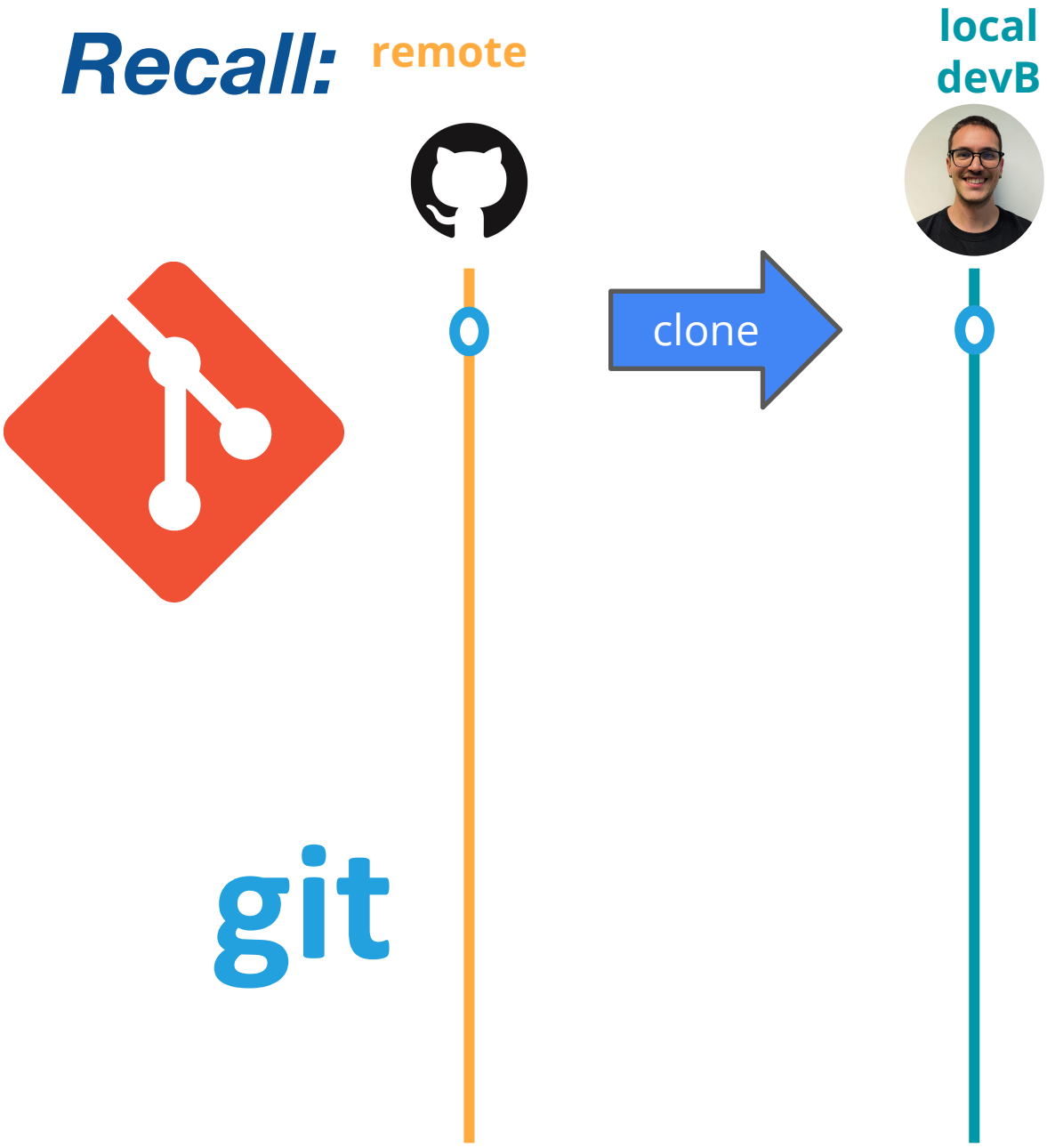
remote

local
devB

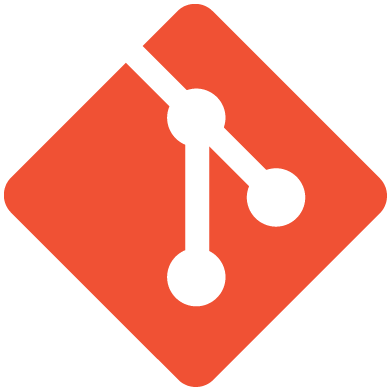


git

Recall:



Recall:



git

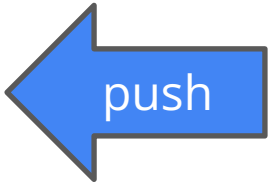
remote

local
devB

Recall:

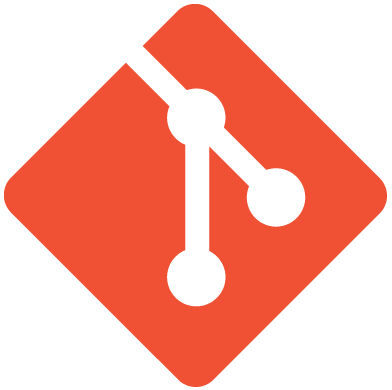
remote

local
devB



git

Recall:



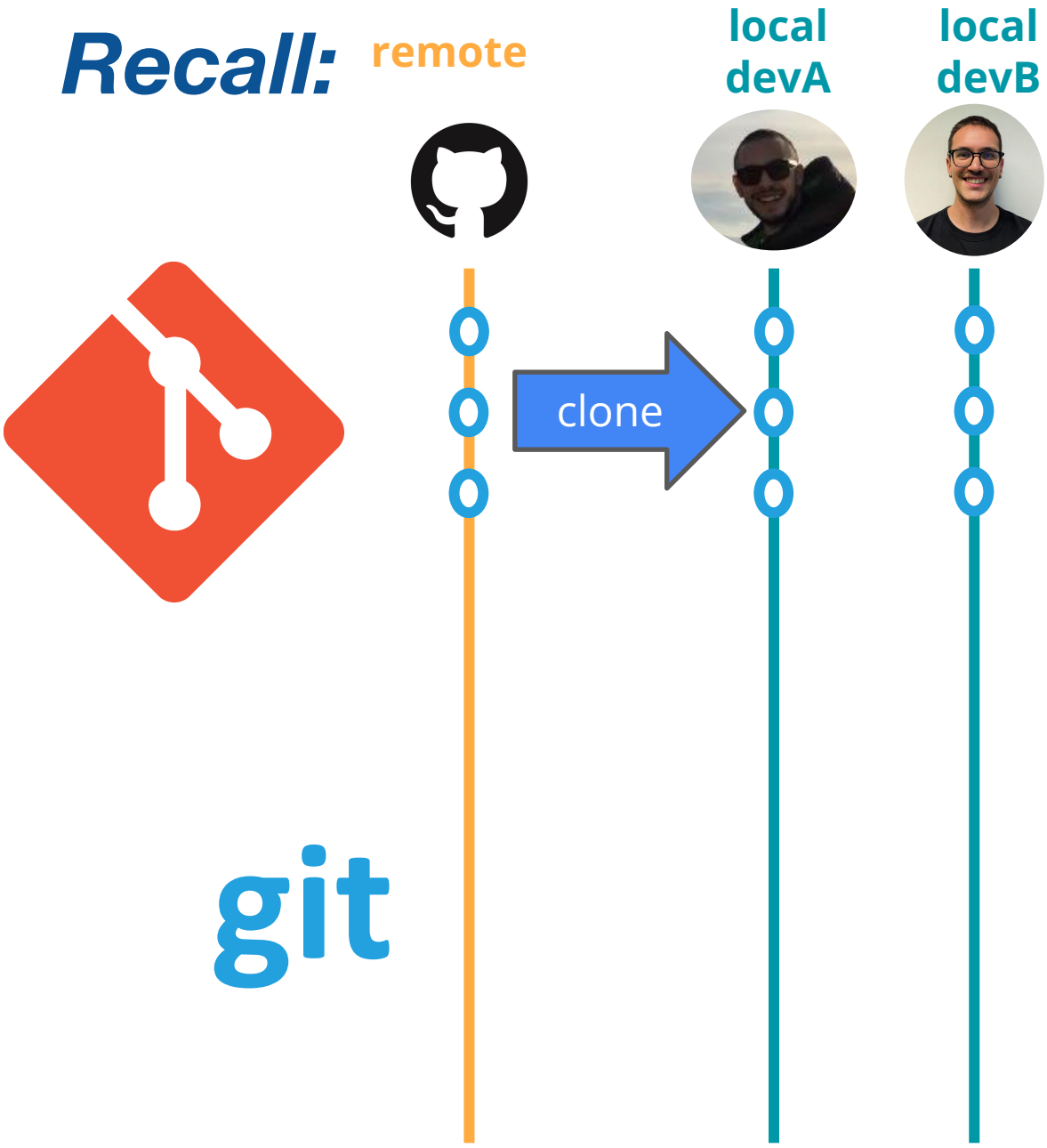
git

remote

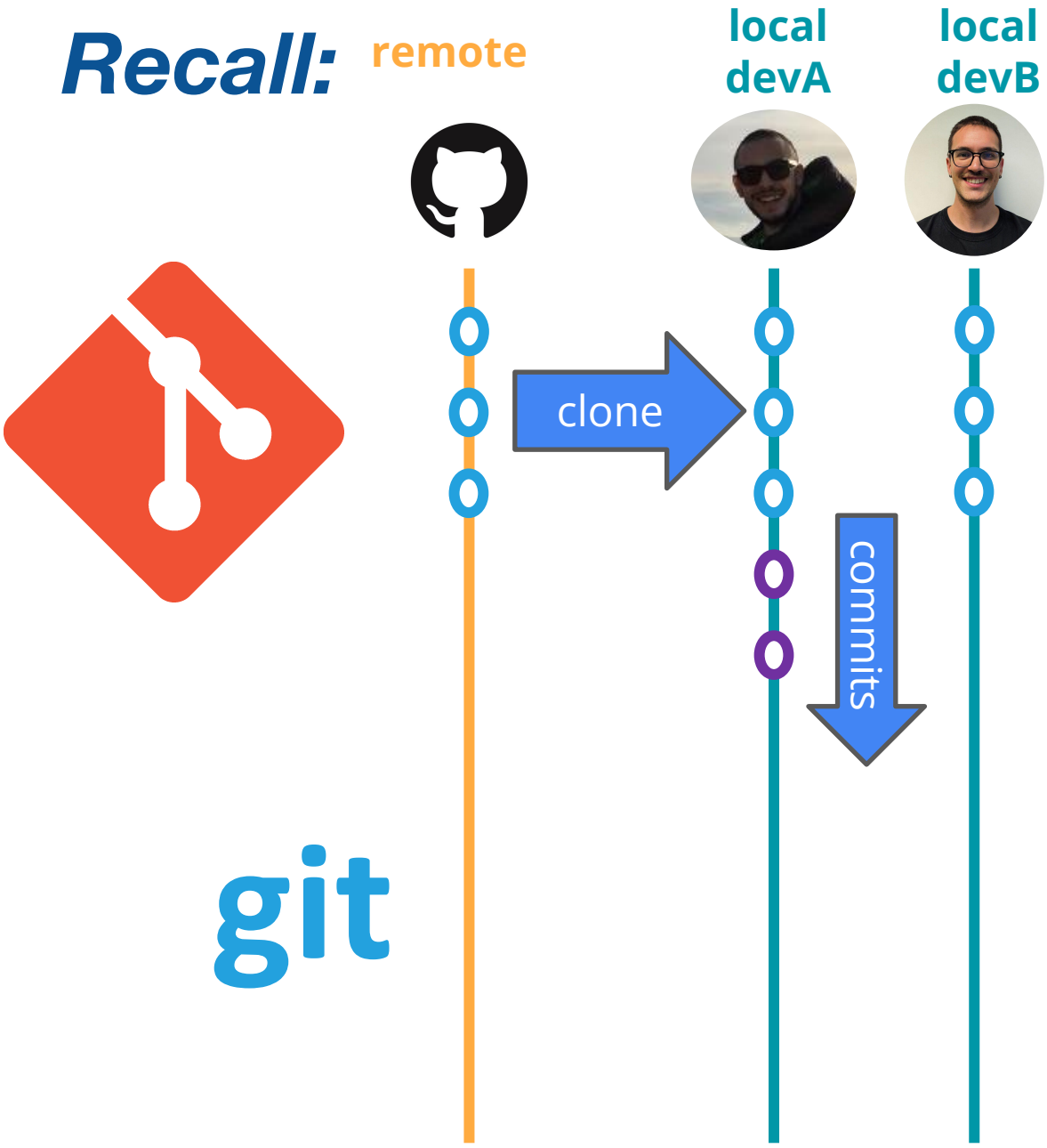
local
devA

local
devB

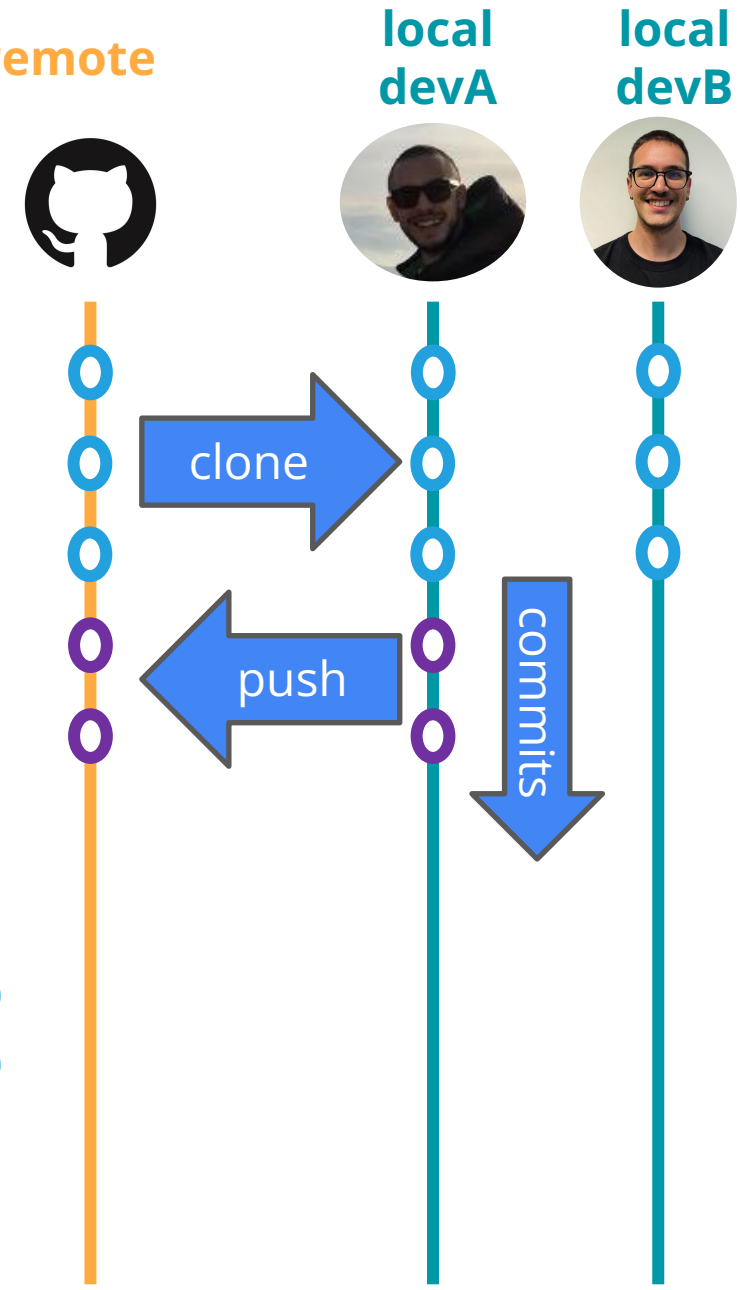
Recall:



Recall:

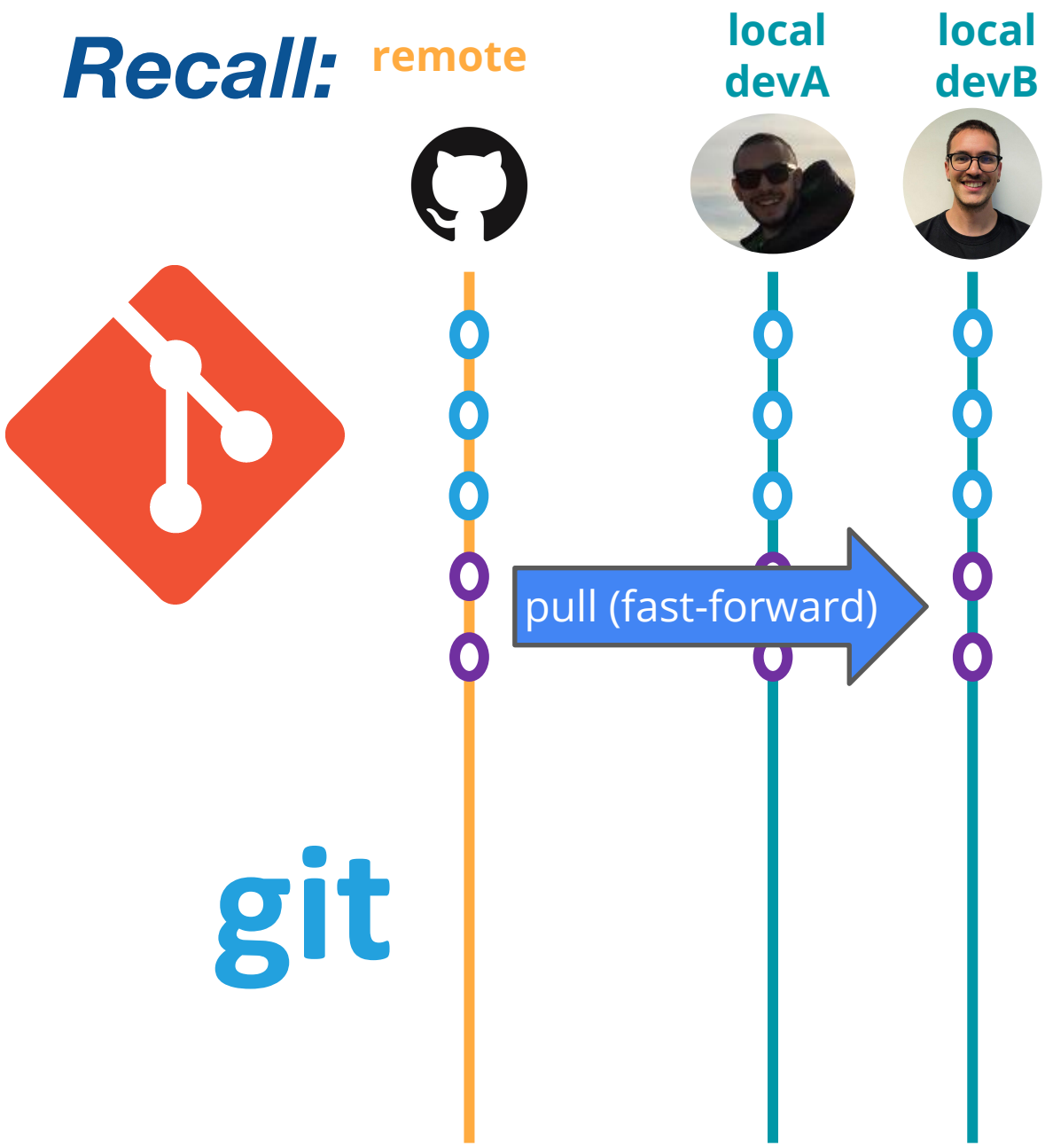


Recall:

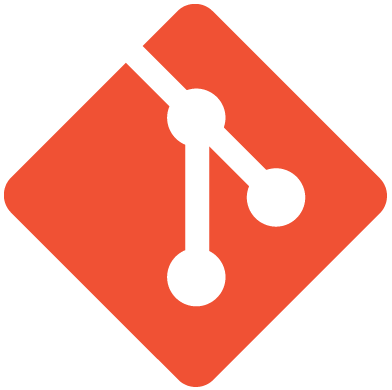


git

Recall:



Recall:



git

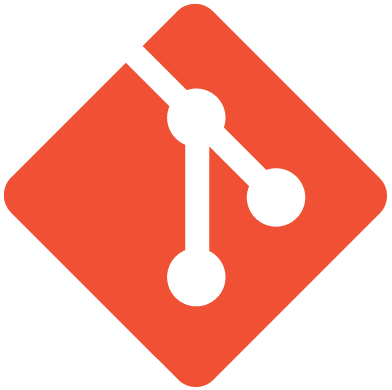
remote

local
devA

local
devB

local
devC

Recall:



git

remote

local
devA

local
devB

local
devC

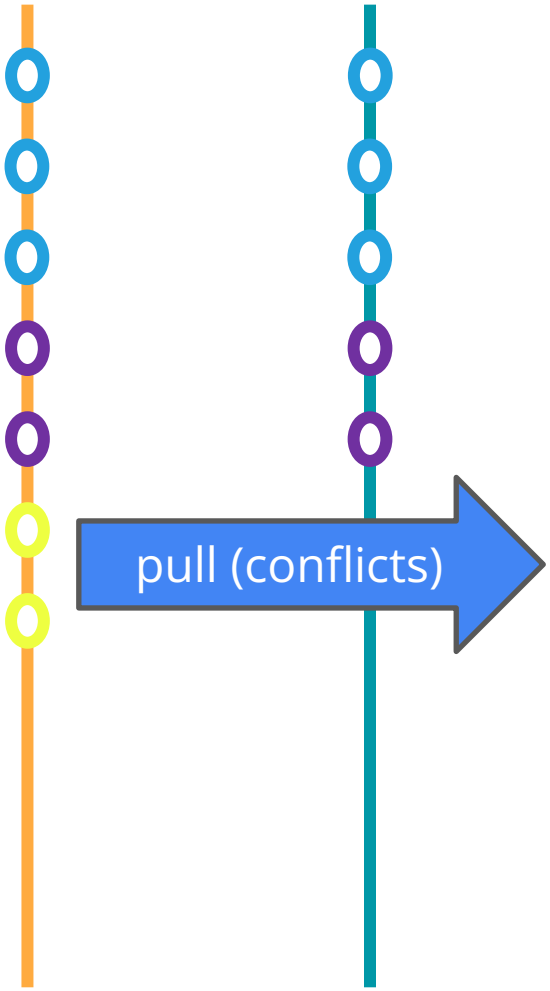
Recall:

remote

local
devA

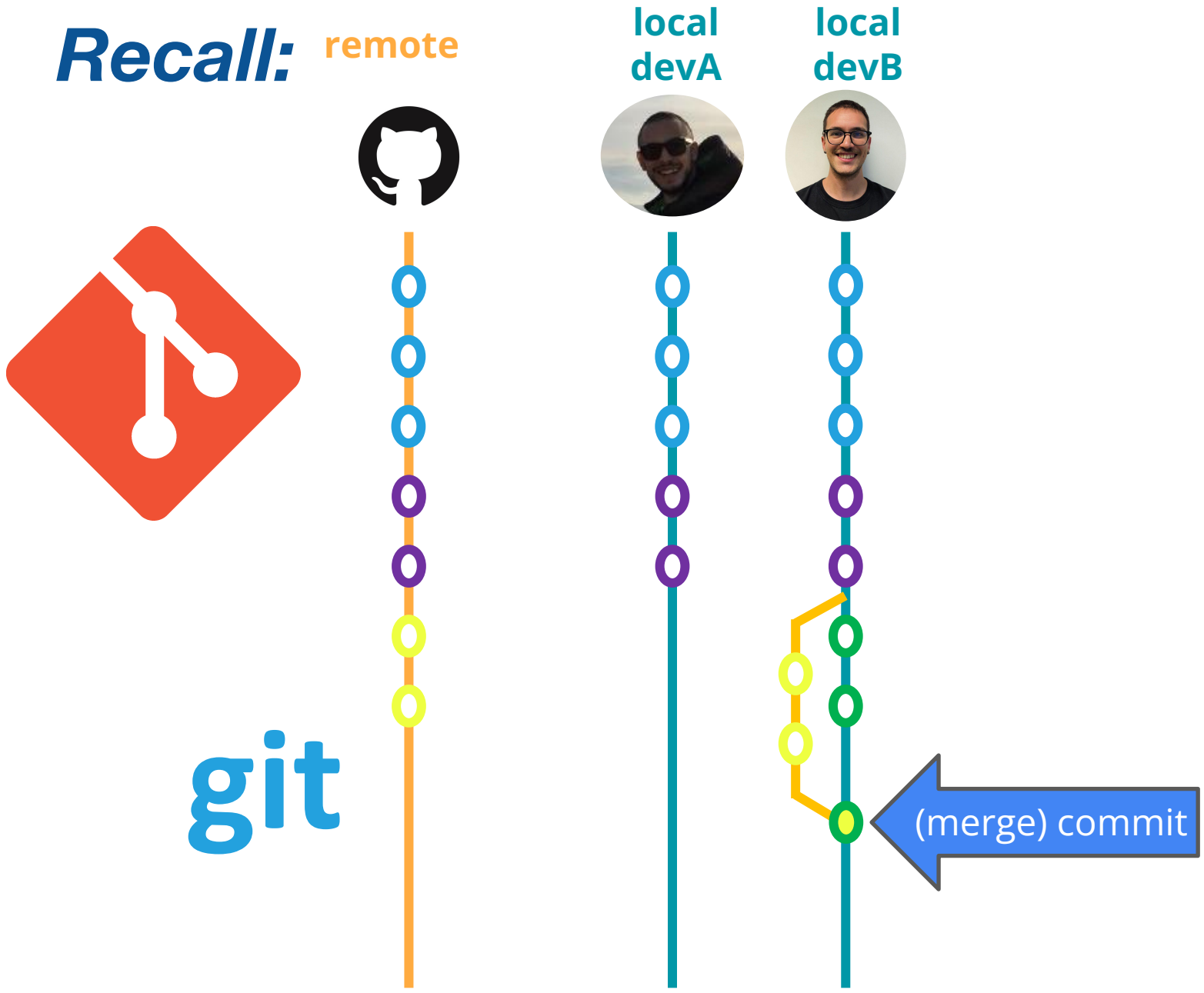
local
devB

local
devC

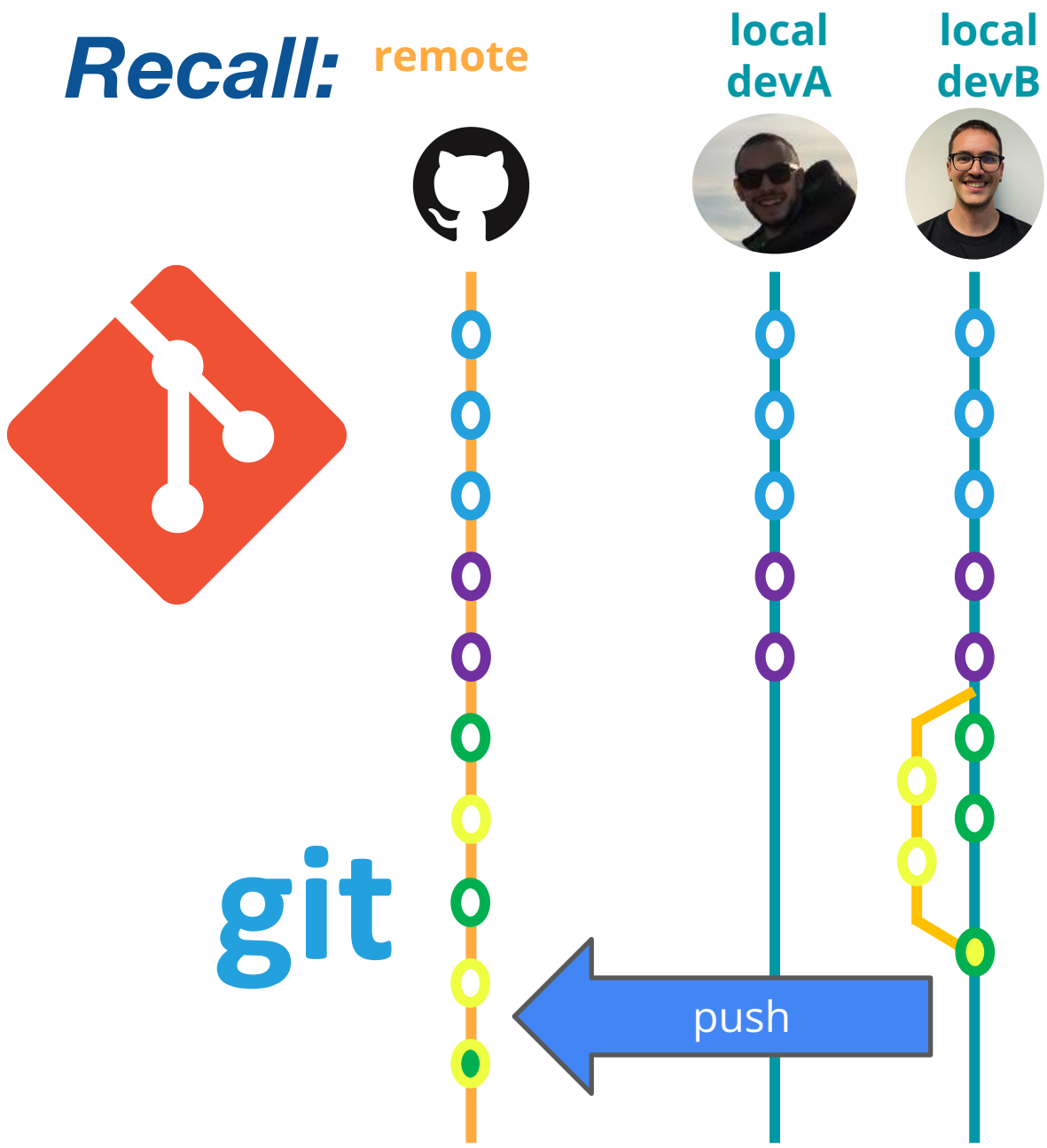


git

Recall:



Recall:



Recall: L'idea e definizione Algoritmo

Algoritmo

- (idea) Descrizione della soluzione di problema scritta in modo da poter essere eseguita da un esecutore (eventualmente diverso dall'autore dell'algoritmo)

- aka (= definizione di algoritmo) Sequenza finita di passi atomici, non ambigui che operano su dati per arrivare alla soluzione del problema

Recall: L'idea e definizione Algoritmo e Programma

Algoritmo

- (idea) Descrizione della soluzione di problema scritta in modo da poter essere eseguita da un esecutore (eventualmente diverso dall'autore dell'algoritmo)

- aka (= definizione di algoritmo) Sequenza finita di passi atomici, non ambigui che operano su dati per arrivare alla soluzione del problema

Programma

- Algoritmo scritto in modo da poter essere eseguito da un calcolatore (esecutore automatico)

Come realizzare un algoritmo

Parte 1/4: Capire il problema

- Quale e' il problema generale che si scerca di r

Come realizzare un algoritmo

Parte 2/4: Fare/creare un piano

- Ci possono essere diverse strategie per risolvere lo stesso problema
  - Ipotizzare e verificare
  - Cercare dei *pattern*
  - Risolvere problemi più piccoli
  - Disegnare uno schema

Come realizzare un algoritmo

Parte 3/4: Portare avanti il piano

- Mettere in azione il vostro piano!
- Rimanere sul piano deciso a meno che non vi siano evidenti motivi per credere che esso non funzionerà più

La pazienza è il vostro miglior alleato

Come realizzare un algoritmo

Parte 4/4: Ragionare e comprendere

- Comprendere quello che si è fatto e dove l'algoritmo individuato possa essere applicato al meglio

La pratica è fondamentale!

TESTARE IL PROPRIO LAVORO

Se vi è qualche cosa che non funziona nel vostro programma/algoritmo, gli utenti lo **troveranno**!

Today Objectives

1. ~~Recap + Challenge discussion~~
2. ~~Refresh on Algorithms~~
3. ~~Dev with P5: Live Editor and Local VScode~~
4. ~~Draw with P5~~
5. ~~Variables + JS arithmetic~~
6. ~~Repetitions~~
7. ~~Branches~~
8. ~~Put everything live and check github!~~

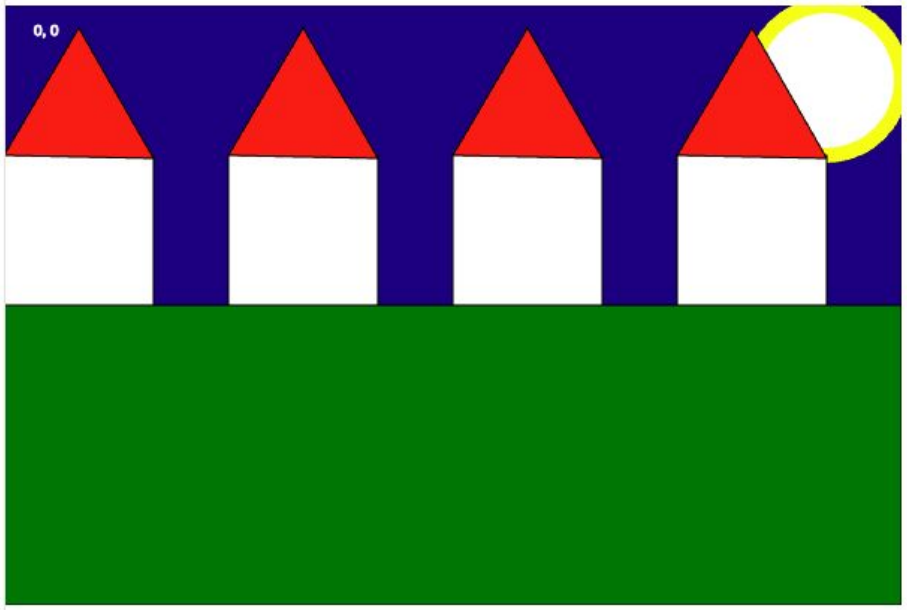
https://editor.p5js.org/

Lecture 2 Challenge

Prepare a drawing of something you would like to draw.
Example below.

USE VARIABLES, LOOPS and Arithmetics and whatsoever :D
(add the link to the tab:
https://docs.google.com/spreadsheets/d/1Ykz1MleWlT3YBjkBwBtLP5mb76Tm7z604qrcklUBCwU/edit?usp=drive_link)

Deadline: Monday 7th October 12:00



https://pixabay.com/photos/candle-light-the-background-fire-7562931/

Thank you for your attention

Davide Conficconi <davide.conficconi@polimi.it>
Alessandro Nazzari <alessandro.nazzari@polimi.it>

Acknowledgements

Everything already cited in the slides
Part of this material comes from:

- LCG- IA 22-23; 23/24 edition and their corresponding Acks, especially I. Di Dio Lavore
- https://thecodingtrain.com
- https://p5js.org/tutorials/
- The book on the right :)