WEB DEVELOPMENT

THE POWERFUL JAVASCRIPT HIGHER ORDER METHODS

THE FOLLOWING METHODS ARE:

1. foreach()
2. filter()
3. sort()
4. map()

CONSIDER AN EXAMPLE

```
const product = [
    {id: 1, category: "IPhone", name:"IPhone11Pro", start: 2007 , price: 80000},
    {id: 2, category: "MI", name:"Note8", start: 2008, price: 20000},
    {id: 3, category: "Samsung", name:"GalaxyS20", start: 2002 , price: 30000},
    {id: 4, category: "OnePlus", name:"OnePlus8", start: 2005, price: 50000},
    {id: 5, category: "Nokia", name:"Nokia6.2", start: 2001, price: 10000},
    {id: 6, category: "IPhone", name:"IPhoneSE", start: 2007 , price: 80000},
    {id: 7, category: "MI", name:"Note7", start: 2008, price: 20000},
    {id: 8, category: "Samsung", name:"GalaxyS10", start: 2002 , price: 30000},
    {id: 9, category: "OnePlus", name:"OnePlus8Pro", start: 2005, price: 50000},
    {id: 10, category: "Nokia", name:"Nokia6", start: 2001, price: 10000}
]
```

1. FOREACH

Foreach loop (or for each loop) is a control flow statement for traversing items in a collection.

```
1  //forLoop
2  for(let i = 0; i<product.length; i++){
3      console.log("[Product]", product[i]);
4  }
5
6  //forEach
7  product.forEach((pro) => {
8      console.log("[Product]",pro)
9  })
```

```
[Product] ▶ {id: 1, category: "IPhone", name: "IPhone11Pro", start: 2007, price: 80000}
[Product] ▶ {id: 2, category: "MI", name: "Note8", start: 2008, price: 20000}
[Product] ▶ {id: 3, category: "Samsung", name: "GalaxyS20", start: 2002, price: 30000}
[Product] ▶ {id: 4, category: "OnePlus", name: "OnePlus8", start: 2005, price: 50000}
[Product] ▶ {id: 5, category: "Nokia", name: "Nokia6.2", start: 2001, price: 10000}
[Product] ▶ {id: 6, category: "IPhone", name: "IPhoneSE", start: 2007, price: 80000}
[Product] ▶ {id: 7, category: "MI", name: "Note7", start: 2008, price: 20000}
[Product] ▶ {id: 8, category: "Samsung", name: "GalaxyS10", start: 2002, price: 30000}
[Product] ▶ {id: 9, category: "OnePlus", name: "OnePlus8Pro", start: 2005, price: 50000}
[Product] ▶ {id: 10, category: "Nokia", name: "Nokia6", start: 2001, price: 10000}
```

2. FILTER

The filter() method creates a new array with all the elements that pass the test implemented by the callback() function.

```
//forLoop
let filterProduct = [];
for(let i = 0; i<product.length; i++){
    if(product[i].category === "MI"){
        product1.push(product[i])
    }
}
console.log(filterProduct);

//filter
let filterProduct = product.filter(
    pro => pro.category === "MI"
);
console.log(filterProduct);
```

```
index.html:47
▼(2) [{…}, {…}]
  ▶0: {id: 2, category: "MI", name: "Note8", start: 2008, price: 20000}
  ▶1: {id: 7, category: "MI", name: "Note7", start: 2008, price: 20000}
    length: 2
```

3. MAP

The map() method creates a new array with the results of calling a function for every array element.

The map() method calls the provided function once for each element in an array.

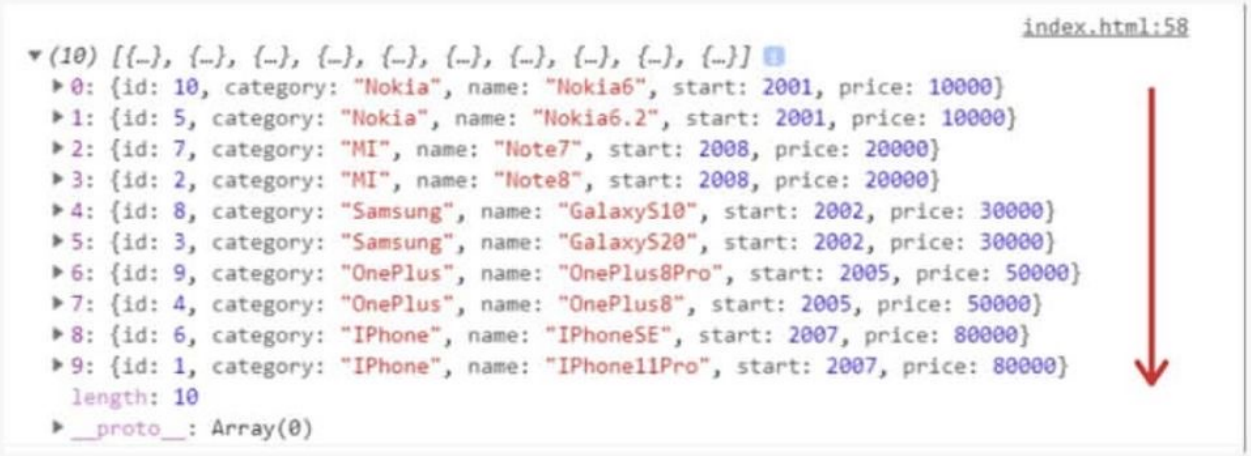
```
1  //map
2  //Create array of product name
3  const productName = product.map((pro) => pro.name);
4  console.log(productName);
```

```
                                                    index.html:52
▸ (10) ["IPhone11Pro", "Note8", "GalaxyS20", "OnePlus8", "Nokia6.2",
  "IPhoneSE", "Note7", "GalaxyS10", "OnePlus8Pro", "Nokia6"]
```

4. SORT

The sort() method is used to sorts the items of an array. It takes two arguments to compare and sort it accordingly.

```
1  //sort
2  const sortProduct = product.sort((a, b) => (
3      a.price > b.price  ?  1 : -1 )
4  );
5  console.log(sortProduct);
```



BEST 5 CSS TRICKS NOBODY TELLS YOU

CENTER ANYTHING IN 3 LINES.

width and height properties would be necessary

```css
.content{
    width: 100%;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}
```

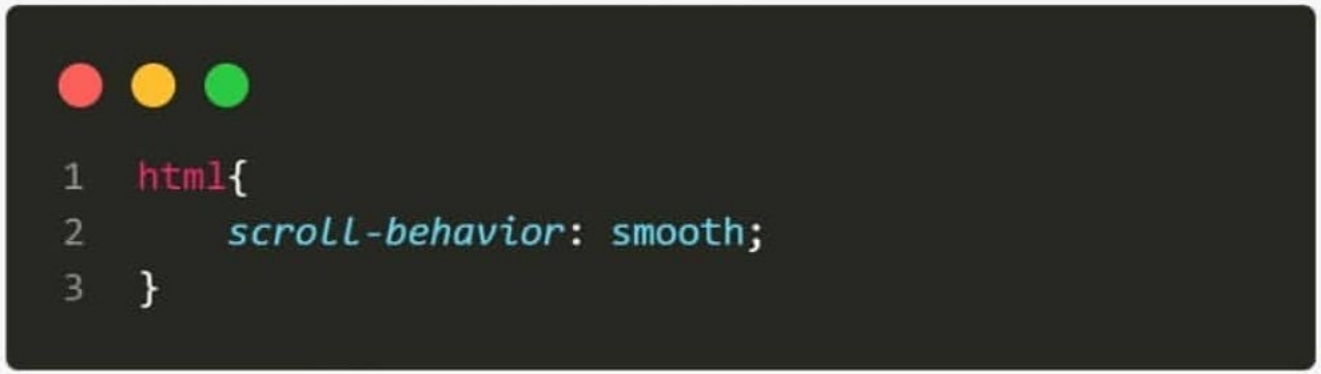
ADD SHADOW TO TRANSPARENT IMAGE

```css
.content img{
    filter: drop-shadow(2px 4px 8px #585858);
}
```

box-shadow

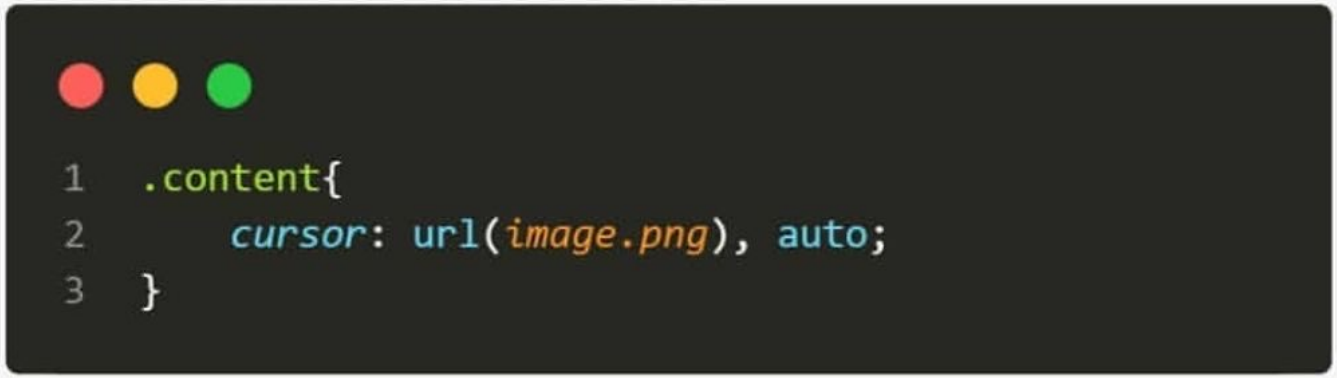
drop-shadow

SMOOTH SCROLLING
WITH ONE LINE OF CODE



```
1  html{
2      scroll-behavior: smooth;
3  }
```

SET IMAGE AS CURSOR
WITH ONE LINE OF CODE



```
.content{
    cursor: url(image.png), auto;
}
```

SET LIMITED CONTENT IN A PARAGRAPH

The -webkit-line-clamp CSS property allows limiting of the contents of a block container to the specified number of lines

```
p{
    -webkit-line-clamp: 3;
}
```

EASY TIPS & TRICKS

BEST WAY TO SEND EMAILS USING HTML TAG

HTML MAILTO ATTRIBUTE

Syntax:

mailto: "email_id"

Code:

```
<a href="mailto:codingbeast9867@gmail.com">Send Mail</a>
```

ADD MULTIPLE EMAIL ID

Syntax:

mailto: "email_id1, email_id2"

Code:

```
<a href="mailto:codingbeast9867@gmail.com,
    parvezinamdar9867@gmail.com">Send Mail</a>
```

ADD EMAIL & CC TOGETHER

Syntax:

mailto: "email_id1?cc=email_id2"

Code:

```
<a href="mailto:codingbeast9867@gmail.com?
    cc=parvezinamdar9867@gmail.com">Send Mail</a>
```

ADD SUBJECT IN EMAIL

Syntax:

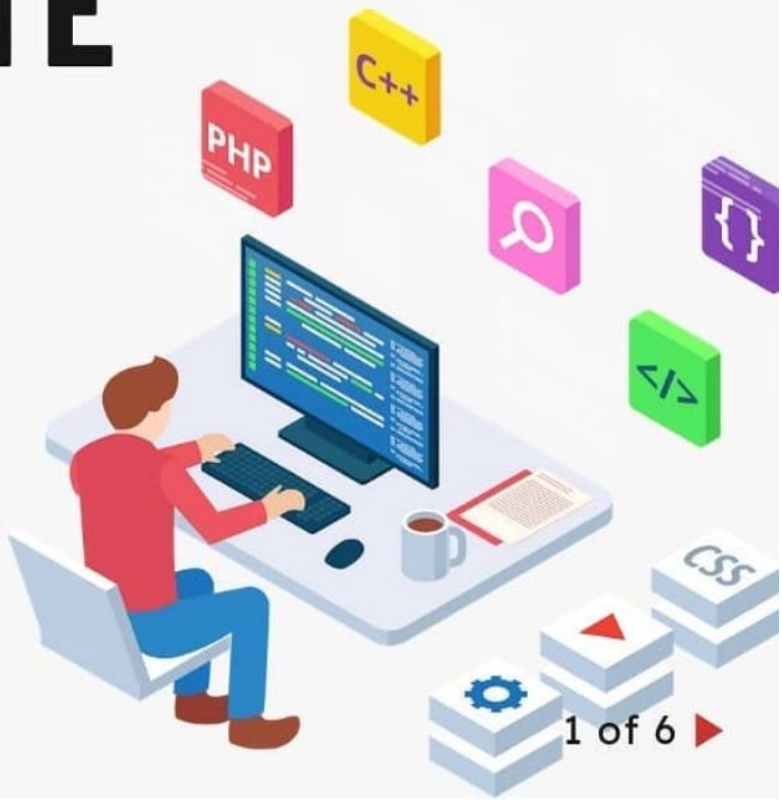
mailto:"email_id1?
cc=email_id2&subject=SUBJECT"

Code:

```
<a href="mailto:codingbeast9867@gmail.com?
    cc=parvezinamdar9867@gmail.com&
    subject=Send Email Using HTML">Send Mail</a>
```

WEB DEVELOPMENT

28 WEBSITE TO KEEP YOURSELF UP TO DATE



NEWS

1. Daily JS
2. HTML5 Weekly
3. Hacker News
4. SlashDot
5. Reddit
6. A list Apart

CHALLENGES

1. Code Combat
2. Code Wars
3. Hacker Rank
4. Hacker Earth
5. Code Chef

TUTORIALS

1. HTML5 Rocks
2. CSS Tricks
3. Smashing Magazine
4. Tuts+
5. Geeks For Geeks
6. W3 School

JOB

1. Fiverr
2. Upwork
3. Instagram
4. LinkedIn
5. Freelancer

NETWORKING

1. Github
2. Coderwall
3. Stack Overflow
4. Geeklist
5. Codepen
6. CSS Community

```
const cookies = "singIn=true; activated=true; os=win10";
//assuming cookies = document.cookie;

const readCookies = (cookie) => {
  let [obj, splitter] = [{}, null];
  let arr = cookie.split(";");

  for (let i = 0; i < arr.length; i++) {
    splitter = arr[i].split("=");
    obj[splitter[0]] = splitter[1];
    splitter = null;
  }
  return obj;
};



readCookies(cookies) ▶{singIn: "true",  activated: "true",  os: "win10"}
```

Jacob Paris 🇨🇦

@jacobmparis

hey everyone I finally solved centering things in CSS

```css
.center {
  text-align: center;
  align-items: center;
  align-self: center;
  align-content: center;
  justify-items: center;
  justify-self: center;
  justify-content: center;
  place-items: center;
  vertical-align: middle;
  line-height: 100%;
  margin: auto;
  position: absolute;
  left: 50%;
  right: 50%;
  transform: translate(-50%, -50%);
}
```

```
const array = ["a","b","c","d","a","c","e","f"];

/**Method 1 using Set*/
const method1 = [... new Set(array)];


/**Method 2 using Object*/
const obj = {}
for(let arr of array){
    obj[arr]=true;
}
const method2 = Object.keys(obj);


/**Method 3 using Filter*/
const method3 = array.filter((ele, index)=>array.indexOf(ele) == index)



/**Method 4 using For loop*/
const method4 = [];

for(let i=0; i<array.length;i++) {
    const arr = array[i];
    method4.indexOf(arr) === -1 && method4.push(arr);
}
```

```
1  function reverseString(str) {
2    const arrayString = Array.from(str);
3    const reversedStringArray = [];
4    for (let i = arrayString.length - 1; i >= 0; i--) {
5      reversedStringArray.push(arrayString[i]);
6    }
7    return reversedStringArray.join("");
8  }
9  reverseString("hello");
```

```
let array = ["A", "B", "C", "Z"]

//This works now
let [last] = array.slice(-1)
let [lastThird] = array.slice(-3)

//This doesn't work yet!
//let last = array.at(-1)

// Z B
console.log(last, lastThird);
```

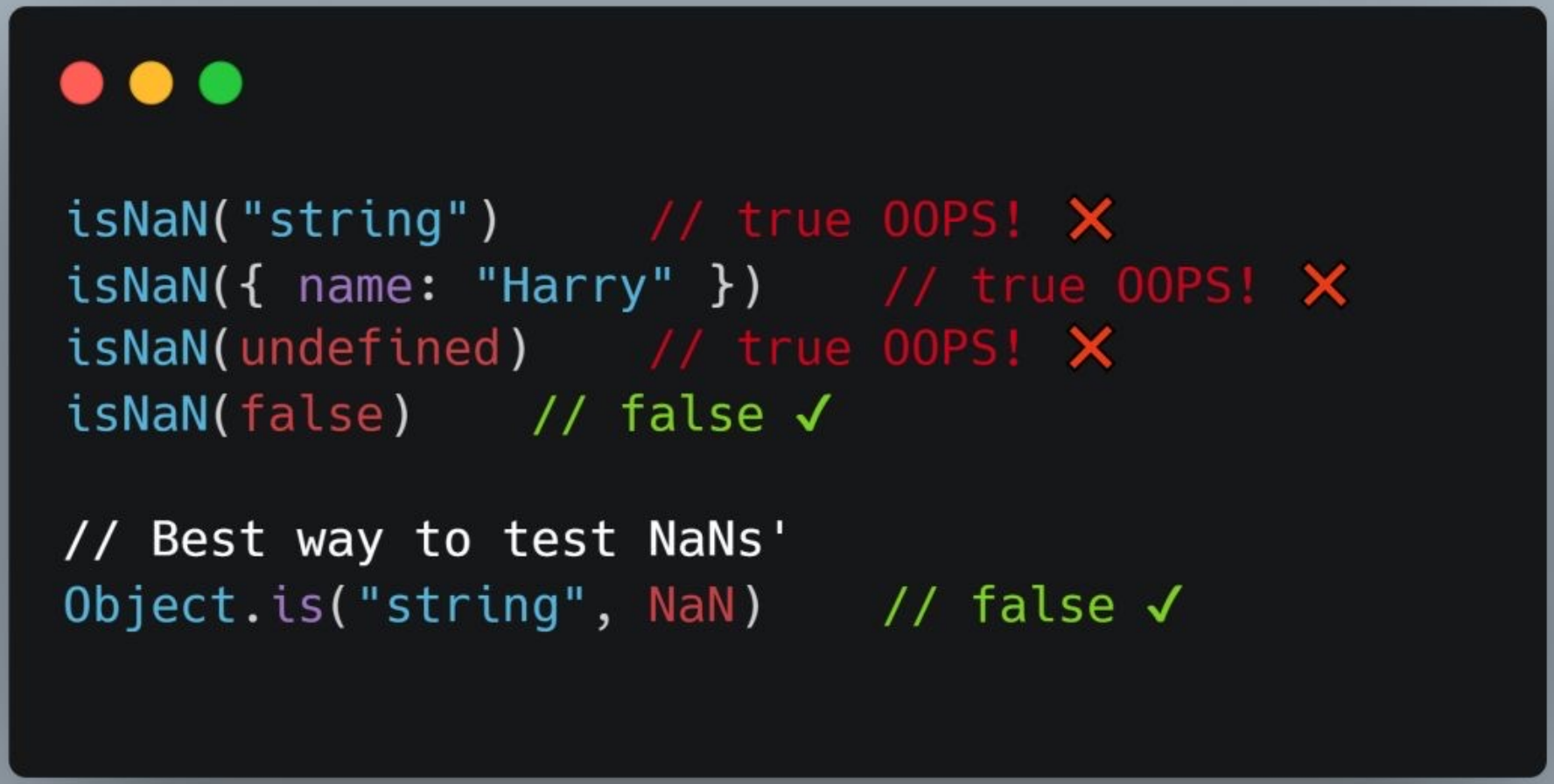
```
let SameNaN = NaN;
console.log(SameNaN != SameNaN);
//true.. This is the best way :p
```

smart-array-access.js

```js
const species = ['octopus', 'squid', 'shark', 'seahorse', 'starfish', 'whale',];

// CONVENTIONAL WAY
const firstItem = species[0];
const lastItem = species[species.length - 1];

// NEW WAY WITH SMART ACCESS
const { 0: firstItem, [species.length - 1]: lastItem } = species;

console.log(firstItem, lastItem);
// outputs: "octopus", "whale";
```

async-operation.js

```
/** BAD, BLOCKS THE EXECUTION */
async function getOptions() {
  // request "toppings" and wait for the response
  // only then goes to the next line
  const toppings = await getToppings();
  const condiments = await getCondiments();
  const buns = await getBuns();


  return { toppings, condiments, buns };
}

/** GREAT, TAKE BENEFIT FROM ASYNC OPS */
async function getOptions() {
  // request all methods asynchronously
  // and only resolves after all methods have either fulfilled or rejected
  const [toppings, condiments, buns] = await Promise.allSettled([
    getToppings(),
    getCondiments(),
    getBuns(),
  ]);

  return { toppings, condiments, buns };
}
```


isNaN("string") // true OOPS! ✗
isNaN({ name: "Harry" }) // true OOPS! ✗
isNaN(undefined) // true OOPS! ✗
isNaN(false) // false ✓
// Best way to test NaNs'
Object.is("string", NaN) // false ✓

Data Types in JS:

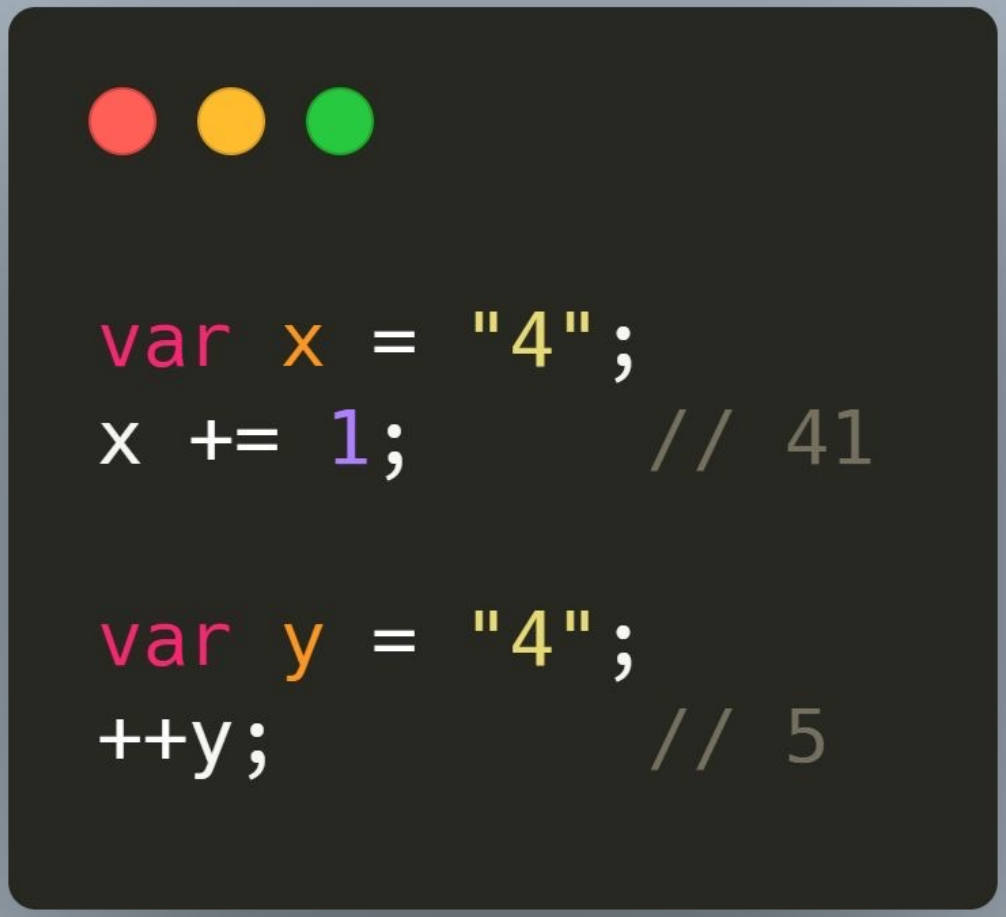
- In JS, **values** have types, but, **variables** do not.

a) Primitive types

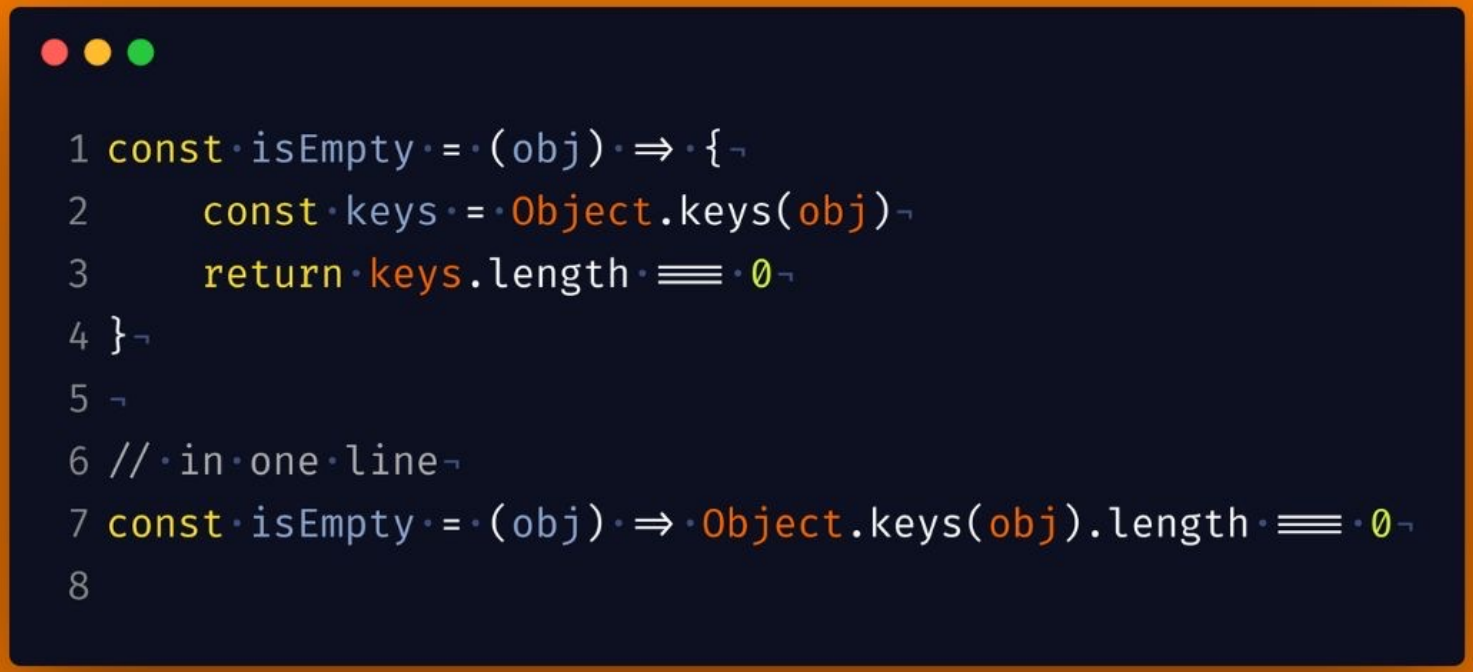
1. undefined
2. string
3. number
4. bigint
5. boolean
6. symbol

b) Non-primitive types

7. object
8. *function "sub-type of object"*
9. *array "sub-type of object"*
10. null
11. undeclared


var x = "4";
x += 1;
// 41
var y = "4";
++y;
// 5

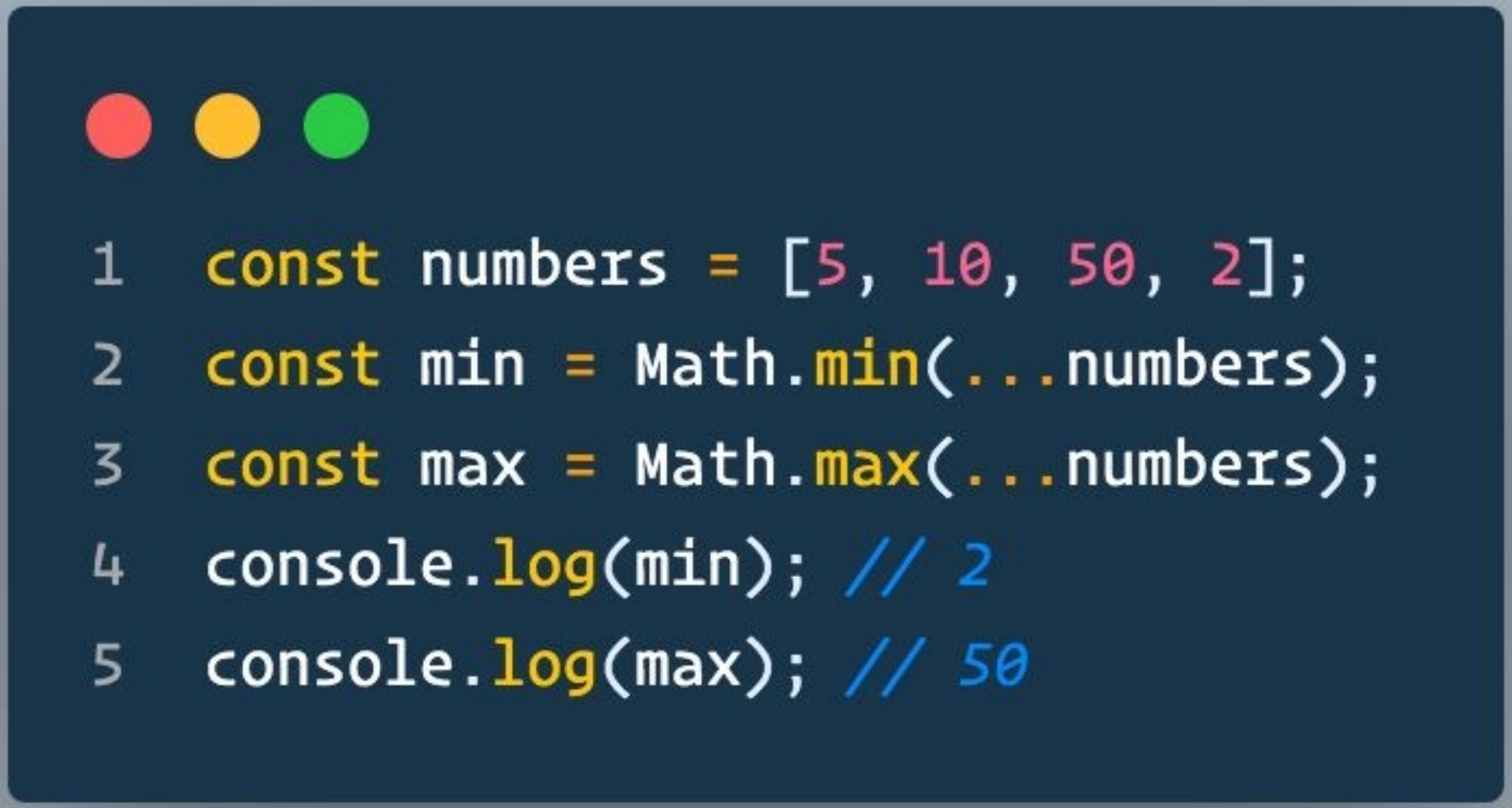
CHECK IF AN OBJECT EMPTY OR NOT



```
const isEmpty = (obj) => {
    const keys = Object.keys(obj)
    return keys.length === 0
}

// in one line
const isEmpty = (obj) => Object.keys(obj).length === 0
```

```
const entries = (obj) => {
    let reversObj = []
    for (const item in obj) {
        reversObj = [
            [item, obj[item]], ...reversObj
        ];
    }

    return reversObj

}

const fromEntries = (array) => {
    let toObject = {}
    for (const arr of array) {
        toObject[arr[0]] = arr[1]
    }
    return toObject
}


const reverse = (input) => {

    if (input.length < 2) {
        return input;
    }
    let reversed = [];

    if (input.constructor === Object) {
        reversed = entries(input);
        return fromEntries(reversed);

    }
    const isNumber = input.constructor === Number
    if (isNumber) {
        input += "";
    }
    for (const char of input) {
        switch (input.constructor) {
            case Array:
                reversed = [char, ...reversed];
                break;
            case String || Number:
                reversed = char + reversed;
                break;
        }
    }
    return isNumber ? +reversed : reversed;

}

reverse({ a: 1, b: 2, c: 3 }); // { c: 3, b: 2, a: 1
reverse([1, 2, 3]);           // [ 3, 2, 1 ]
reverse(123);                // 321
reverse("123");             // "321"
```

```
const split = (string, condition) => {
    let splited = []
    if (condition) {

        let words = ""
        for (let i = 0; i < string.length + 1; i++) {
            if (string[i] === condition || string[i] ===
undefined) {
                splited[splited.length] = words
                words = ""
            } else {
                words += string[i]
            }
        }
    }

    if (!condition) {
        for (const char of string) {
            splited[splited.length] = char

        }
    }
    return splited
}

split("Hello Word!","")
// ["H", "e", "l", "l", "o", " ", "W", "o", "r", "d", "!"]

split("Hello Word!"," ")      // ["Hello", "Word!"]

split("He,ll,o W,or,d!"," ") // ["He", "ll", "o W", "or", "d!"]
```


1 const numbers = [5, 10, 50, 2];
2 const min = Math.min(...numbers);
3 const max = Math.max(...numbers);
4 console.log(min); // 2
5 console.log(max); // 50

Spread Operator

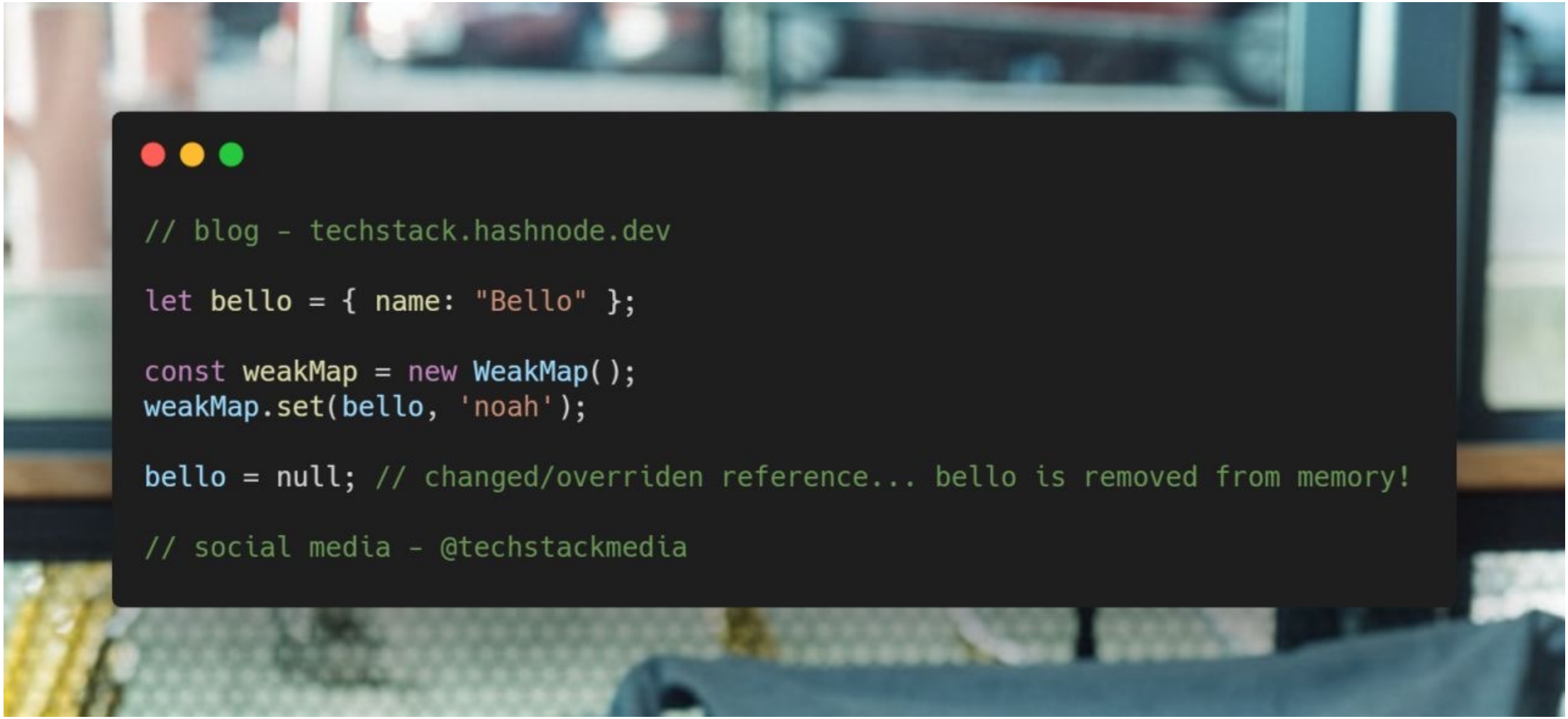
The spread operator is used to replace certain array functions. The spread operator is simply a series of three dots.

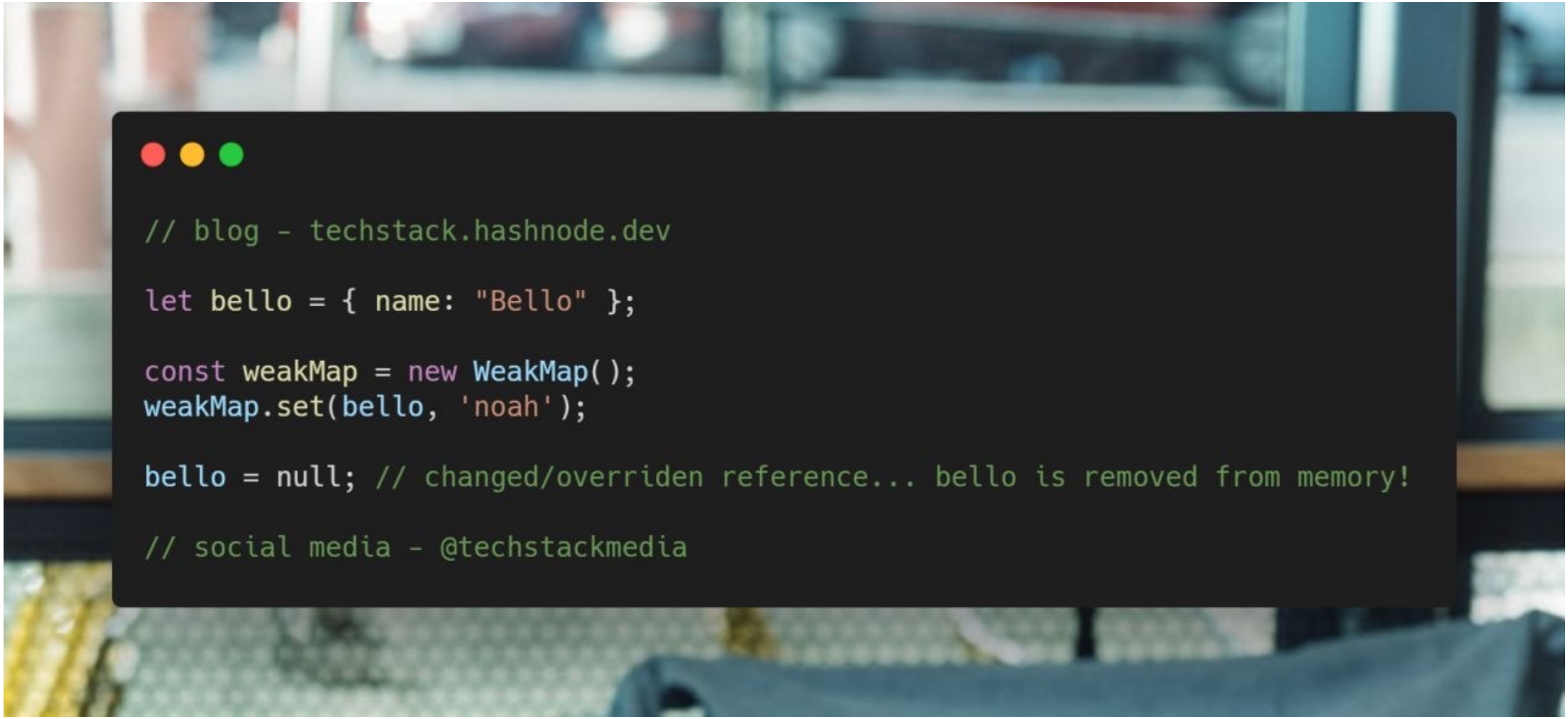

```
//longhand
const odd = [1, 3, 5];
const nums = [2 ,4 , 6].concat(odd);
//shorthand
const odd = [1, 3, 5 ];
const nums = [2 ,4 , 6, ...odd];

console.log(nums); // [ 2, 4, 6, 1, 3, 5 ]
```

multiple conditions

We can store multiple values in the array and we can use the array includes method.

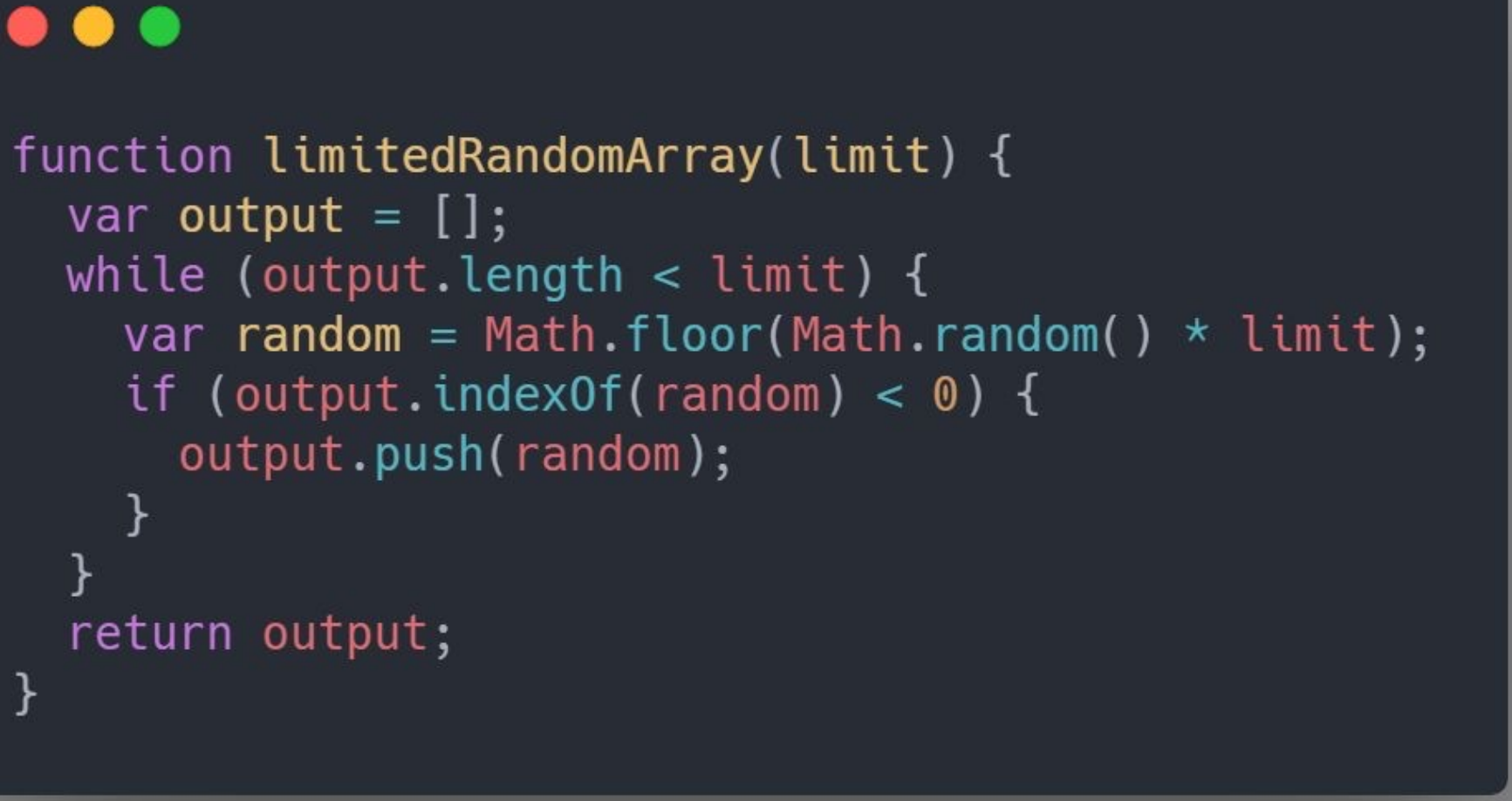
```
//longhand
if (x === 'abc' || x === 'def' || x === 'ghi' || x ==='jkl') {
    //logic
}
//shorthand
if (['abc', 'def', 'ghi', 'jkl'].includes(x)) {
   //logic
}
```


// blog - techstack.hashnode.dev
let bello = { name: "Bello" };
const weakMap = new WeakMap();
weakMap.set(bello, 'noah');
bello = null; // changed/overriden reference... bello is removed from memory!
// social media - @techstackmedia


// blog - techstack.hashnode.dev
let bello = { name: "Bello" };
const weakMap = new WeakMap();
weakMap.set(bello, 'noah');
bello = null; // changed/overriden reference... bello is removed from memory!
// social media - @techstackmedia

```ts
const execCallBack =
    (value: string, callBack: Function = () => { }) => setTimeout(() => {
        console.log(value);
        callBack();
    }, 3e3);

// With Callback
execCallBack(
    'first call',
    () => execCallBack(
        'second callback',
        () => execCallBack(
            'third callback',
            () => execCallBack('fourth & final callback')
        )
    )
);
```

```ts
const execPromise =
    (value: string) => new Promise((resolve, reject) => {
        setTimeout(() => {
            console.log(value);
            resolve(value);
        }, 3e3);
    });

// With Promise
execPromise('first promise resolve')
    .then(() => execPromise('second promise resolve'))
    .then(() => execPromise('third promise resolve'))
    .then(() => execPromise('fourth & final promise resolve'));
```



```
function limitedRandomArray(limit) {
  var output = [];
  while (output.length < limit) {
    var random = Math.floor(Math.random() * limit);
    if (output.indexOf(random) < 0) {
      output.push(random);
    }
  }
  return output;
}
```

DuckDuckGo VS Google

https://jigneshthedeveloper.com/

DuckDuckGo	Google
Most private search engine.	Best search results.
Ease of use.	Personalized results.
Does not track users or save any search results.	Tracks every move you make, even when you leave the search results.
First priority is protecting users' privacy	Sells ads based on user information.
Clean interface.	Multiple highly-functional integrations.

https://jigneshthedeveloper.com/

WEB DEVELOPMENT

6 IMPORTANT TIPS FOR WRITING BETTER CSS

DONT REPEAT YOURSELF

```
<button class="btn primary-btn">Submit</button>
<button class="btn secondary-btn">Submit</button>
```

```css
.primary-btn{
    background-color: #000;
    color: #fff;
    font-size: 15px;
    border-radius: 10px;
    padding: 12px 20px;
}
.secondary-btn{
    background-color: #ff3939;
    color: #fff;
    font-size: 15px;
    border-radius: 10px;
    padding: 12px 20px;
}
```

Don't

```css
.btn{
    color: #fff;
    font-size: 15px;
    border-radius: 10px;
    padding: 12px 20px;
}

.primary-btn{
    background-color: #000;
}

.secondary-btn{
    background-color: #ff3939;
}
```

Do

AVOID !IMPORTANT

When particular style of an element is not changing we use !important tag to override(change) the style of the existing element.

```
button{
    color: #fff !important;
}
```

Don't

And to change that override element we have to again use !important, which is hard to maintain so its important to not use !important😁 in CSS

USE SHORTHANDS

Some properties like padding, margin, border etc. allow you to code in one line which is very simpler to write and also saves a lot of time during development.

```css
.btn{
    padding-top: 10px;
    padding-right: 10px;
    padding-left: 10px;
    padding-bottom: 10px;
}
```

Don't

```css
.btn{
    padding: 10px;
    /* same value in all side */
    padding: 10px 20px;
    /* (top bottom) (right left) */
    padding: 10px 20px 15px;
    /* top (right left) bottom*/
    padding: 10px 20px 15px 25px;
    /* top right bottom left */
}
```

Do

USE FONT FALLBACK

The font-family property should hold several font "fallback" system, to ensure maximum compatibility between browsers. If the browser does not support the first font, it tries the next font.

```
1  body{
2      font-family: Arial;
3  }
```

Don't

Do

```
1  body{
2      font-family: Arial, Helvetica, sans-serif;
3  }
```

USE HEX INSTEAD OF COLOR NAME

It's preferable to use hex value since it is supported by all browsers and We can specify wide range of colors than the named colors.

```
1  p{
2      color: white;
3  }
```

Don't

Do

```
1  p{
2      color: #ffffff;
3  }
```

REDUNDANT PROPERTIES

Writing same code again and again leads to redundancy in code instead use comma(,) to give same styling to one or more selector.

```
#selector-1 {
    font-style: italic;
    color: #e7e7e7;
    margin: 5px;
    padding: 20px
}
.selector-1 {
    font-style: italic;
    color: #e7e7e7;
    margin: 5px;
    padding: 20px
}
```

Don't

```
#selector-1, .selector-1 {
    font-style: italic;
    color: #e7e7e7;
    margin: 5px;
    padding: 20px
}
```

Do

```
const sum = function (a) {
  return function (b) {
    if (b) {
      return sum(a + b);
    }
    return a;
  };
};

const sum1 = function (a) {
  return function (b) {
    return b ? sum1(a + b) : a;
  };
};

const sum2 = (a) => {
  return (b) => {
    return b ? sum2(a + b) : a;
  };
};

const sum4 = (a) => (b) => b ? sum4(a + b) : a;

console.log(sum4(1)(2)(3)(4)(5)());
```

```
function mergeTwoSortedArrays(leftArray, rightArray) {
  let result = [];
  var leftIndex = 0;
  var rightIndex = 0;

  while (leftIndex < leftArray.length && rightIndex < rightArray.length) {
    if (leftArray[leftIndex] < rightArray[rightIndex]) {
      result.push(leftArray[leftIndex]);
      leftIndex++;
    } else {
      result.push(rightArray[rightIndex]);
      rightIndex++;
    }
  }

  return [
    ...result,
    ...leftArray.slice(leftIndex),
    ...rightArray.slice(rightIndex),
  ];
}

const leftArray = [5, 20, 30, 45, 100];
const rightArray = [10, 25, 28, 40, 42];
let output = mergeTwoSortedArrays(leftArray, rightArray);
```

```
const romanToInteger = function (romanString) {
  const romanNumerals = { I: 1, V: 5, X: 10, L: 50, C: 100, D: 500, M: 1000 };

  return [...romanString]
    .reverse()
    .reduce(
      (accumulator, currentValue, index, original) =>
        !index ||
        romanNumerals[currentValue] >= romanNumerals[original[index - 1]]
          ? accumulator + romanNumerals[currentValue]
          : accumulator - romanNumerals[currentValue],
      0
    );
};

console.log(romanToInteger("III")) // 3
console.log(romanToInteger("IV")) // 4
console.log(romanToInteger("IX")) // 9
console.log(romanToInteger("LVIII")) // 58
console.log(romanToInteger("MCMXCIV")) // 1994
console.log(romanToInteger("DCXXI")) // 621
```

```
const date1 = new Date("2010-06-02");
const date2 = new Date("2010-06-02");

console.log(date1 === date2); // false
console.log(date1.toString() === date2.toString()); // true
console.log(date1.valueOf() === date2.valueOf()); // true
```

Code snippet 1 - var is global scoped, In this example Only one variable i is created in the Heap memory and is reused. By the time setTimeout is executed i has already incremented to 6. So 6 is printed 5 times.

Code snippet 2 - let is block scoped, In this example for every loop a new variable with its own value is created in the Heap memory. By the time setTimeout is executed, we have 5 different variable values in Heap memory. So all values one by one are printed like 1 2 3 4 5.

#var #let #variables #scope #javascript #webdevelopment #frontenddevelopment #uidevelopment

```
for (var i = 1; i <= 5; i++) {
  setTimeout(() => {
    console.log(i);
  }, i * 1000);
}
// Output - 6 6 6 6 6

for (let i = 1; i < 6; i++) {
  setTimeout(() => {
    console.log(i);
  }, i * 1000);
}
// Output - 1 2 3 4 5
```

JOIN US & LEARN
WEB DEVELOPMENT

justify-content

flex-flow

align-items

PART-1

CSS FLEXBOX

flex-basis

SWIPE

flex-direction

flex-wrap

JOIN US & LEARN
WEB DEVELOPMENT

justify-content

flex-flow

align-items

PART-1

CSS FLEXBOX

flex-basis

SWIPE

flex-direction

flex-wrap

FLEXBOX

TO USE FLEXBOX WE WILL USE DISPLAY PROPERTY.
LET'S FIRST SEE THE OUTPUT WITHOUT FLEX.

HTML

```
<div class="box">1</div>
<div class="box">2</div>
<div class="box">3</div>
<div class="box">4</div>
```

CSS

```
.box {
  width: 120px;
  height: 120px;
  background: #0d1117;
  color: orange;
  margin: 15px;
  display: grid;
  place-items: center;
  font-size: 40px;
  font-weight: bolder;
}
```

NOW LET'S APPLY FLEX-DIRECTION

SAVE THIS POST

DIRECTION

JOIN US & LEARN
WEB DEVELOPMENT

```
body {
  display: flex;
}
```


```
body {
  display: flex;
  flex-direction: row | column | row-reverse | column-reverse;
}
```

flex-direction: row;

flex-direction: row-reverse;

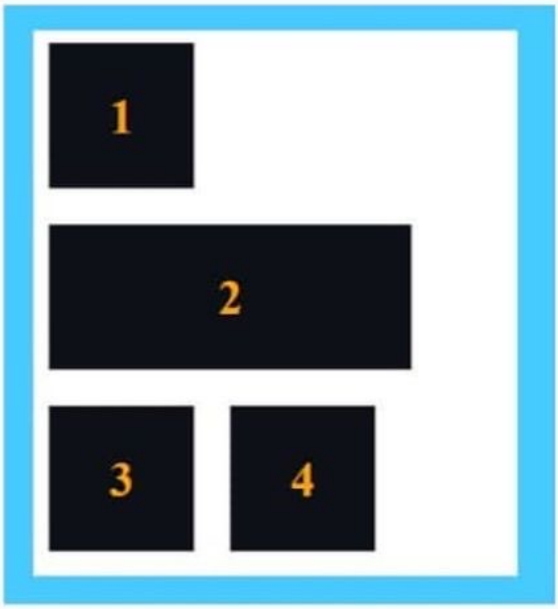
SAVE THIS POST

WRAP

JOIN US & LEARN
WEB DEVELOPMENT

flex-wrap: wrap;

```css
.container {
  display: flex;
  flex-wrap: wrap;
}
.box:nth-child(2) {
  width: 300px;
}
```

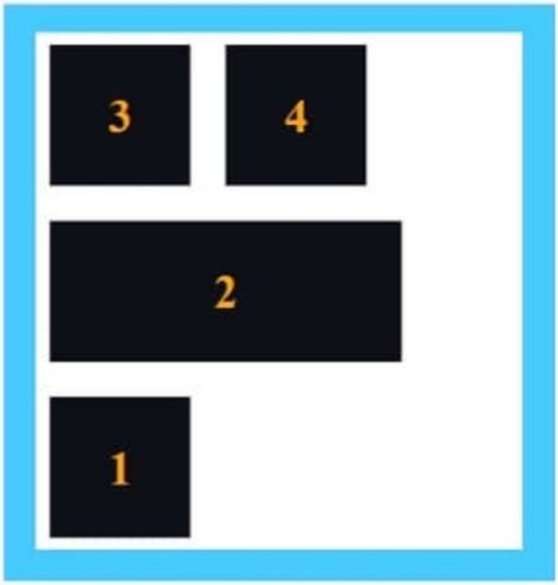


flex-wrap: nowrap;

```css
.container {
  display: flex;
  flex-wrap: nowrap;
}
.box:nth-child(2) {
  width: 300px;
}
```

flex-wrap: wrap-reverse;

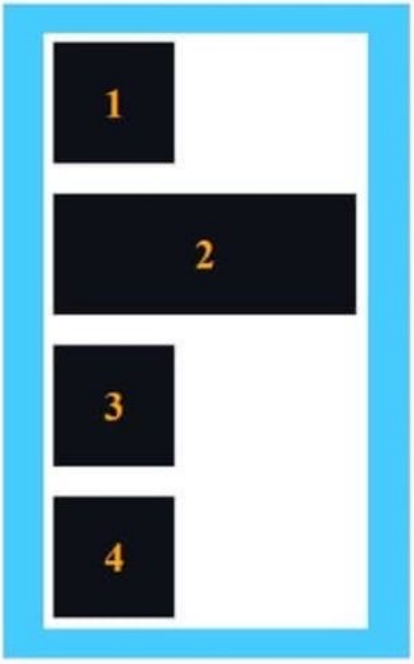
```css
.container {
  display: flex;
  flex-wrap: wrap-reverse;
}
.box:nth-child(2) {
  width: 300px;
}
```



flex-flow

This is a shorthand for the **flex-direction** and **flex-wrap** properties together.

```
.container {
  display: flex;
  flex-flow: column wrap;
}
```



The default value is **row nowrap.**

JUSTIFY-CONTENT

SAVE THIS POST

JUSTIFY-CONTENT

JOIN US & LEARN
WEB DEVELOPMENT

```
.container {
  display: flex;
  justify-content: flex-start | flex-end | center | space-between | space-around | space-evenly;
}
```

flex-start

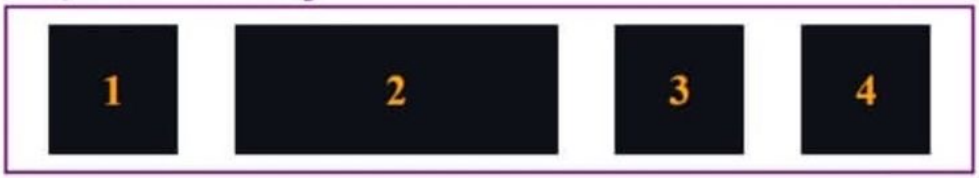
flex-end

center

space-between

space-around

space-evenly



align-items

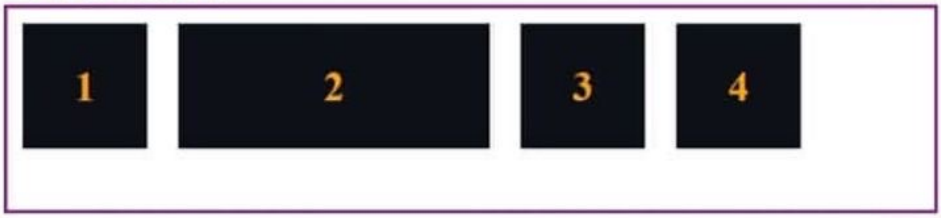
SAVE THIS POST

align-items

JOIN US & LEARN
WEB DEVELOPMENT

```
.container {
  display: flex;
  align-items: stretch | flex-start | flex-end | center | baseline;
}
```

flex-start



flex-end

center

stretch

baseline

SAVE & SWIPE

SAVE THIS POST

JOIN US & LEARN
WEB DEVELOPMENT

IN THIS PART WE DISCUSSED:-

- FLEX-DIRECTION
- FLEX-WRAP
- FLEX-FLOW
- JUSTIFY-CONTENT
- ALIGN-ITEMS

IN PART-2 WE WILL DISCUSS :-

- ORDER
- FLEX-GROW
- FLEX-SHRINK
- FLEX-BASIS
- ALIGN-SELF

5 BASIC STEPS TO CONVERT YOUR WEBSITE FROM LIGHT TO DARK MODE

WEB DEVELOPMENT

HERE YOU WILL ALSO LEARN THE POWER OF ROOT VARIABLE IN CSS

ADD A DATA ATTRIBUTE IN HTML TAG

```
<html lang="en" data-theme="light">
```

You can give any name to the data attribute for eg: data-mode, data-abc here its data-theme

Setting data attribute = "light" will allow you to have a light mode effect in your website

- BUT HOW? -

ADD A ROOT VARIABLE FOR LIGHT AND DARK MODE IN CSS

```css
html[data-theme='light'] {
    --bg: #f5f6f7;
    --color-text: #9c9c9c;
    --color-shadow: -4px -2px 4px 0px #ffffff,
                    4px 2px 6px 0px #DFE4EA;
}

html[data-theme='dark'] {
    --bg: #131419;
    --color-text: #fff;
    --color-shadow: -3px -3px 10px rgba(255,255,255,0.05),
                    3px 3px 15px rgba(0,0,0,0.5);
}
```

Root variable allow you to make a global variable which you can use anywhere in your CSS code, you can define any variable you want for eg: color, background, border, shadow etc.

ADD CHECKBOX IN HTML

```
<input type="checkbox" name="theme">
```

Checkbox will help you to toggle from light mode to dark mode or vice versa.

Note: In Javascript, there are many ways to call checkbox, but here i will call it using name = "theme"

CALL CHECKBOX IN JS AND ADD AN "CHANGE EVENT"

```
let checkbox = document.querySelector('input[name=theme]');
checkbox.addEventListener('change',function(){
    if(this.checked){
        document.documentElement.setAttribute('data-theme','dark');
    }else{
        document.documentElement.setAttribute('data-theme','light');
    }
})
```

In these code, we are checking whether the checkbox is checked or not , if checked it will change the data attribute of html tag to dark otherwise it will set it to light.

If checked: <html data-theme = "dark" else "light">

CALL THE ROOT VARIABLE

```
body{
    background: var(--bg);
    color: var(--color-text);
    width: 100%;
    height: 100vh;
}
.card{
    width: 100%;
    height: 100vh;
    background: var(--bg);
    color: var(--color-text);
    box-shadow: var(--color-shadow);
}
```

In these code, var("root variable name") is use to call a root variable.

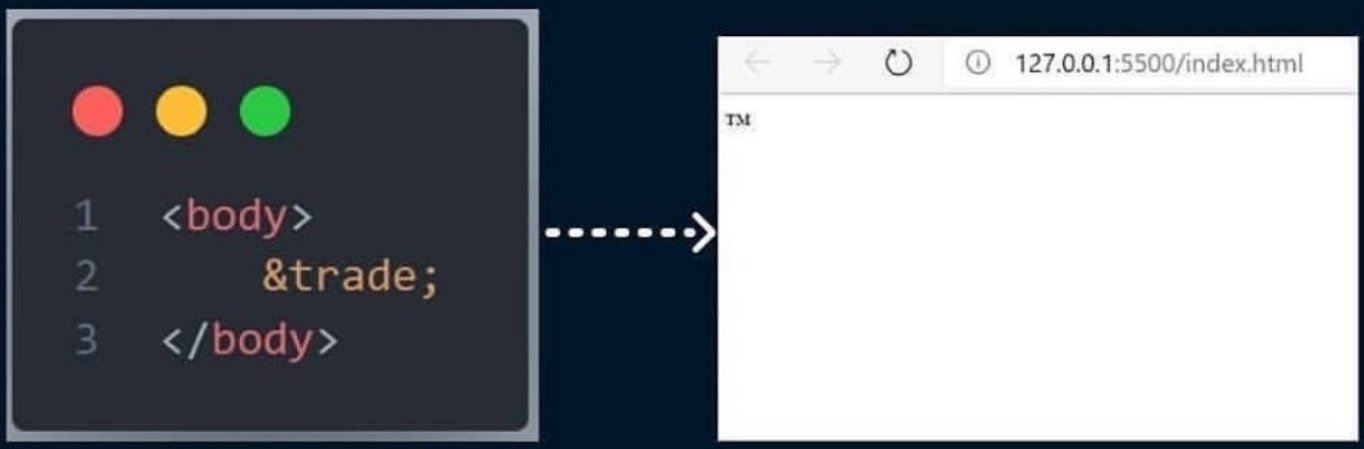
Note: You can call root variable anywhere using var().

SPECIAL SYMBOLS IN HTML

CHEATSHEET

There are some special symbols, which can't be typed using keyboard.

But we can add them to our website using some codes, like this....



CHEATSHEET

GREEK SYMBOLS

SYMBOL	NAME	CODE
α	Alpha	α
β	Beta	β
γ	Gamma	γ
δ	Delta	δ
ε	Epsilon	ε
θ	Theta	θ
λ	Lambda	λ
μ	Mu	μ
π	Pi	π

TRADE SYMBOLS

SYMBOL	NAME	CODE
™	Trademark	™
®	Registered trademark	®
©	Copyright	©

CARD SYMBOLS

SYMBOL	NAME	CODE
♥	Hearts	♥
♦	Diamond	♦
♠	Spades	♠
♣	Clubs	♣

CURRENCY SYMBOLS

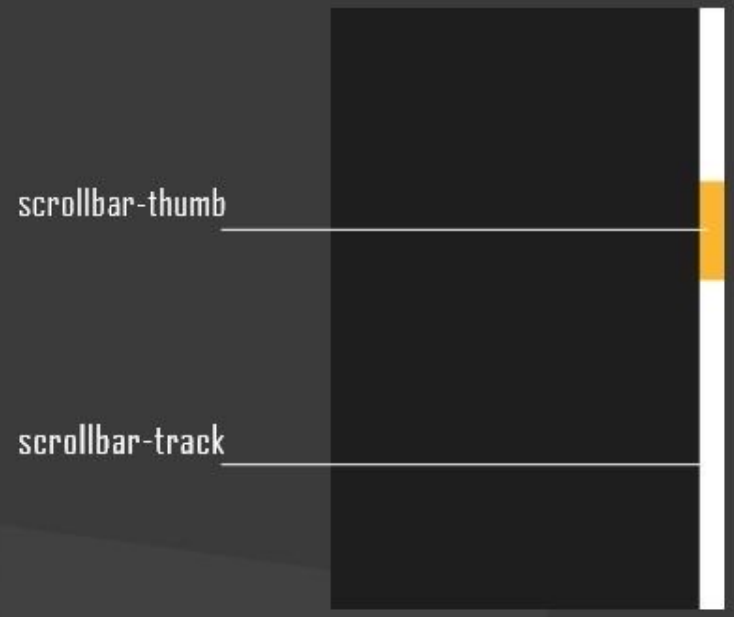
SYMBOL	NAME	CODE
₹	Indian Rupee	&#8377;
€	Euro	&euro;
₱	Peso	&#8369;
₣	French Franc	&#8355;
฿	Bitcoin	&#8383;
₩	Won	&#8361;
₽	Ruble	&#8381;
₾	Lari	&#8382;

MATH SYMBOLS

SYMBOL	NAME	CODE
÷	Divide	÷
↑	Up arrow	↑
→	Right arrow	→
↓	Down arrow	↓
←	Left arrow	←
√	Square root	√
∝	Proportion	∝
∫	Integral	∫
∴	Therefore	∴
⇒	Implies	⇒
∀	For all	∀

how to create a custom scrollbar

```css
/* width */
::-webkit-scrollbar {
    width: 10px;}

/* Track */
::-webkit-scrollbar-track {
    background: #FFF;}

/* Handle */
::-webkit-scrollbar-thumb {
    background: #FCB62E;}

/* Handle on hover */
::-webkit-scrollbar-
thumb:hover {
    background: #f8bf4e;}
```



Jaafar Ali

JS TIP

SORT AN ARRAY OF INTEGERS

```
const numbers = [40, 100, 1, 5, 25, 10];
numbers.sort(); // [ 1, 10, 100, 25, 40, 5 ]
```

```
const numbers = [40, 100, 1, 5, 25, 10];
const copyNumbers = new Int32Array(numbers);
copyNumbers.sort(); // [ 1, 5, 10, 25, 40, 100 ]
```

```
const numbers = [40, 100, 1, 5, 25, 10];
const ascSortedNumbers = [...numbers].sort((a, b) => a - b);
// [ 1, 5, 10, 25, 40, 100 ]
const descSortedNumbers = [...numbers].sort((a, b) => b - a);
// [ 100, 40, 25, 10, 5, 1 ]
```

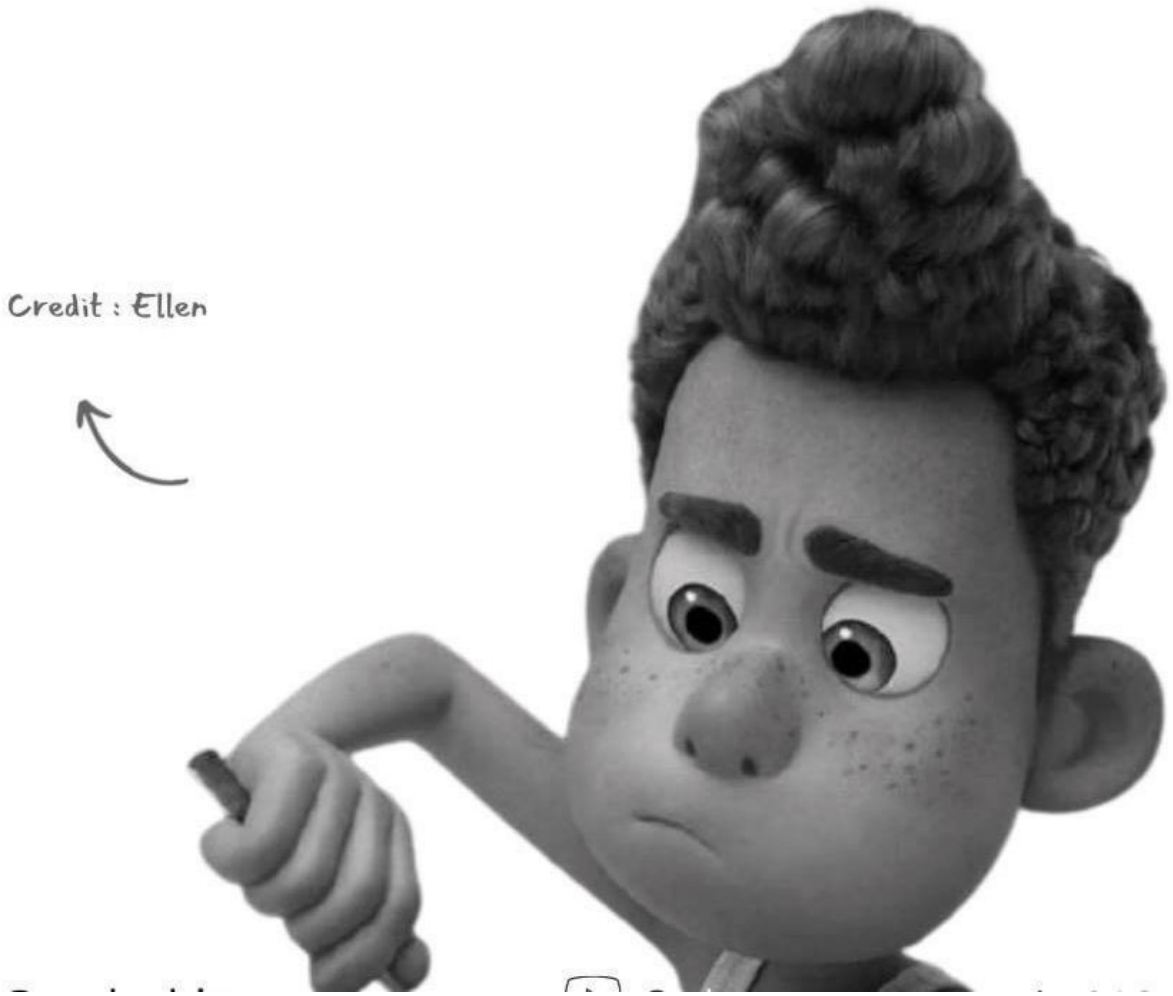
Daniele Serfilippi

JAVASCRIPT QUESTION

```
const person = {
  name: "Ashutosh",
  surname: "Dubey",
  fullName: function () {
    console.log(this.name + " " + this.surname);
  },
  age: function (value) {
    const y = function (value) {
      console.log(value);
      console.log(this.name);
    };
    y(value);
  },
};

// person.fullName();
person.age(5);
```

output ?

WHAT IS A DOCKER ?



Development

Lets say You created an Application

And that's working fine in your machine

Production

But in Production it doesn't work properly

Developers experince it a lot


That is when the Developer's famous words are spoken
It works on my machine
Your application is not working

The Reason could be due to :

- Dependencies
- Libraries and versions
- Framework
- OS Level features
- Microservices

That the developers machine has but not there in the **production environment**

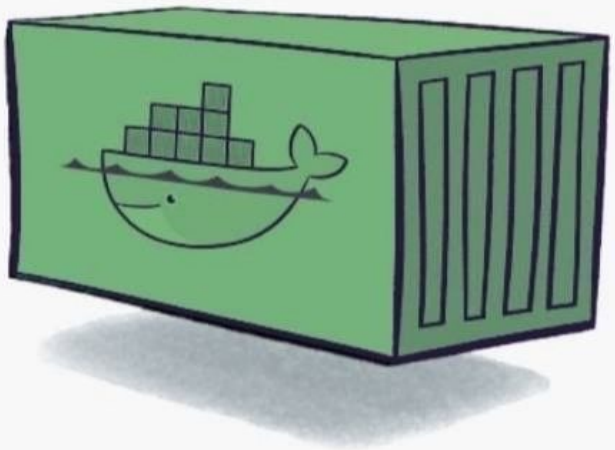
DOCKER

We need a standardized way to package the application with its **dependencies** and deploy it on any environment.

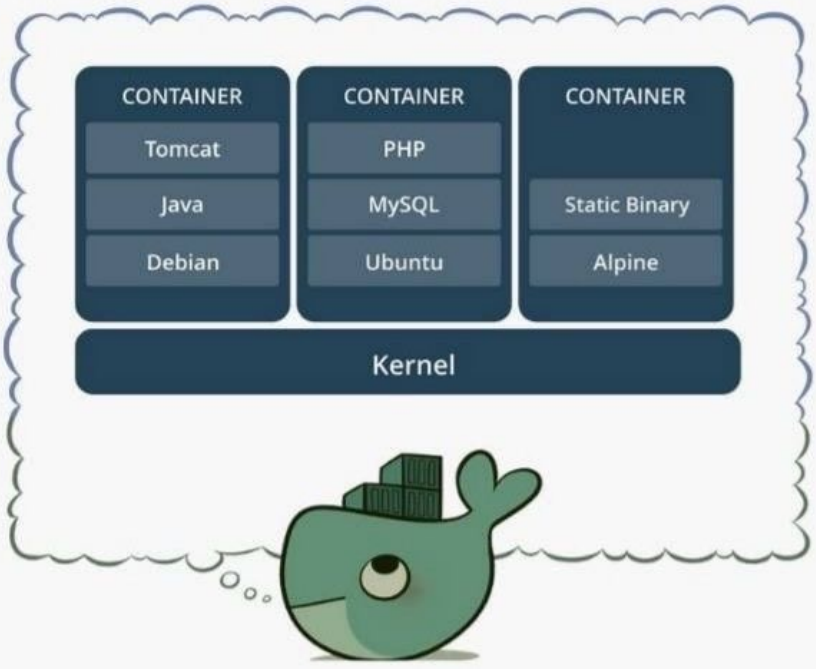
Docker is a tool designed to make it easier to **create**, **deploy**, and **run** applications by using **containers**.

How Does Docker Work?

Docker packages an application and all its **dependencies** in a **virtual container** that can run on any Linux server.



Each container runs as an **isolated process** in the user space and take up **less space** than regular VMs due to their layered architecture.



So it will always work the same regardless of its environment

--> Destructuring nested properties in JavaScript <--

```
// Here is a person Object
const person = {
  email: "gaurav@devjunction.in",
  profile: {
    name: "Gaurav Sharma",
    age: "100",
    experiences: {
      educator: "1 year",
      developer: "2 years"
    }
  }
}

// Here we are destructuring nested 'email', 'name'
and 'developer'
const {
  email,
  profile: {
    name,
    experiences: {
      developer
    }
  }
} = person

console.log(email)
// "gaurav@devjunction.in"

console.log(name)
// "Gaurav Sharma"

console.log(developer)
// "2 years"
```

JS TIP

IMPERATIVE vs DECLARATIVE

HOW

```
function add (numbers) {
  let result = 0;
  for (let i = 0; i < numbers.length; i++) {
    result += numbers[i];
  }
  return result;
}
```

```
function double (numbers) {
  const result = [];
  for (let i = 0; i < numbers.length; i++) {
    result.push(numbers[i] * 2);
  }
  return result;
}
```

WHAT

```
function add (arr) {
  return arr.reduce((prev, curr) => prev + curr);
}
```

```
function double (arr) {
  return arr.map((item) => item * 2);
}
```

Daniele Serfilippi

JAVASCRIPT FUNCTIONS 3 WAYS

- Function Declaration

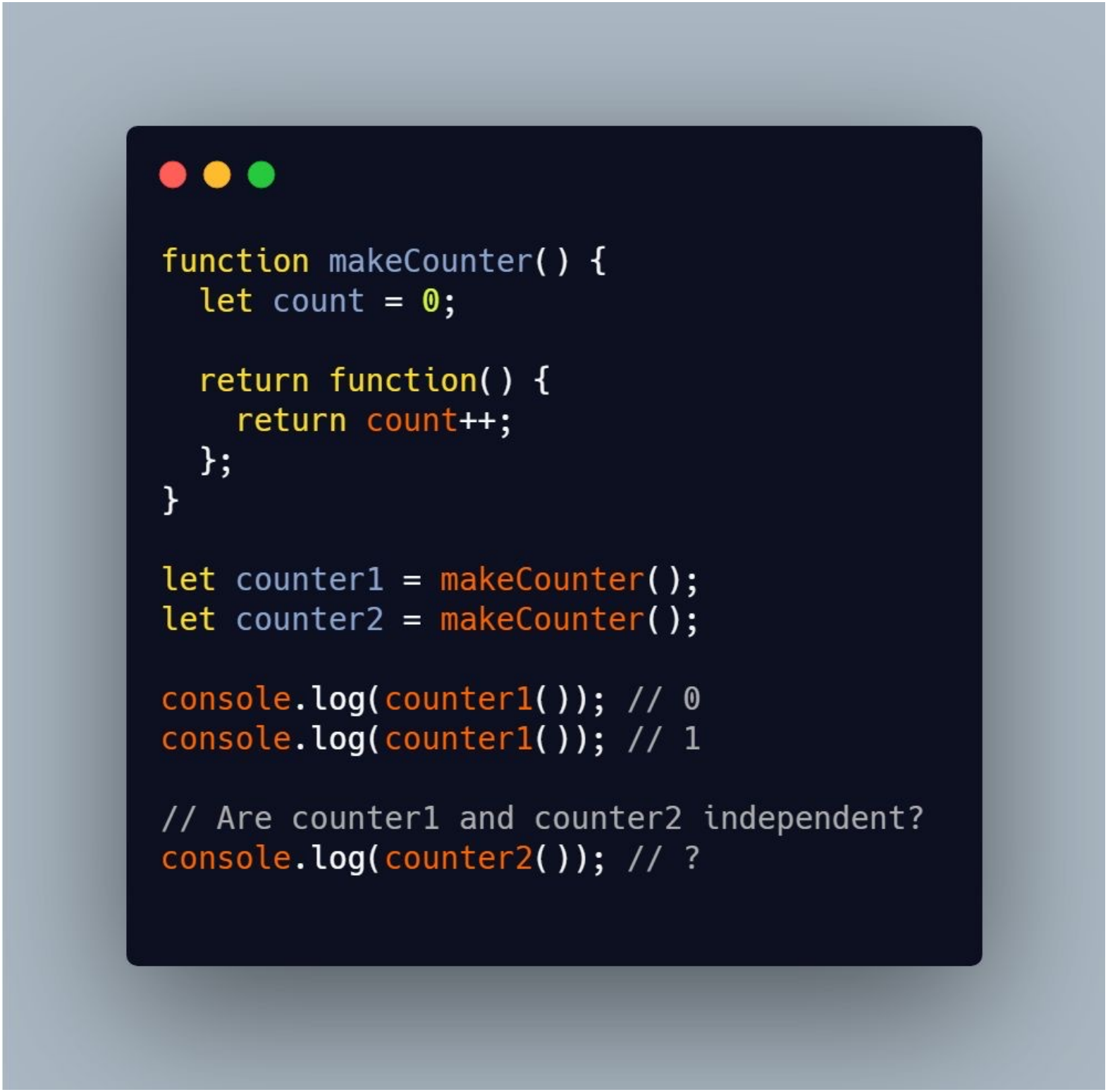
```
function square(x) {
  return x * x
}
```

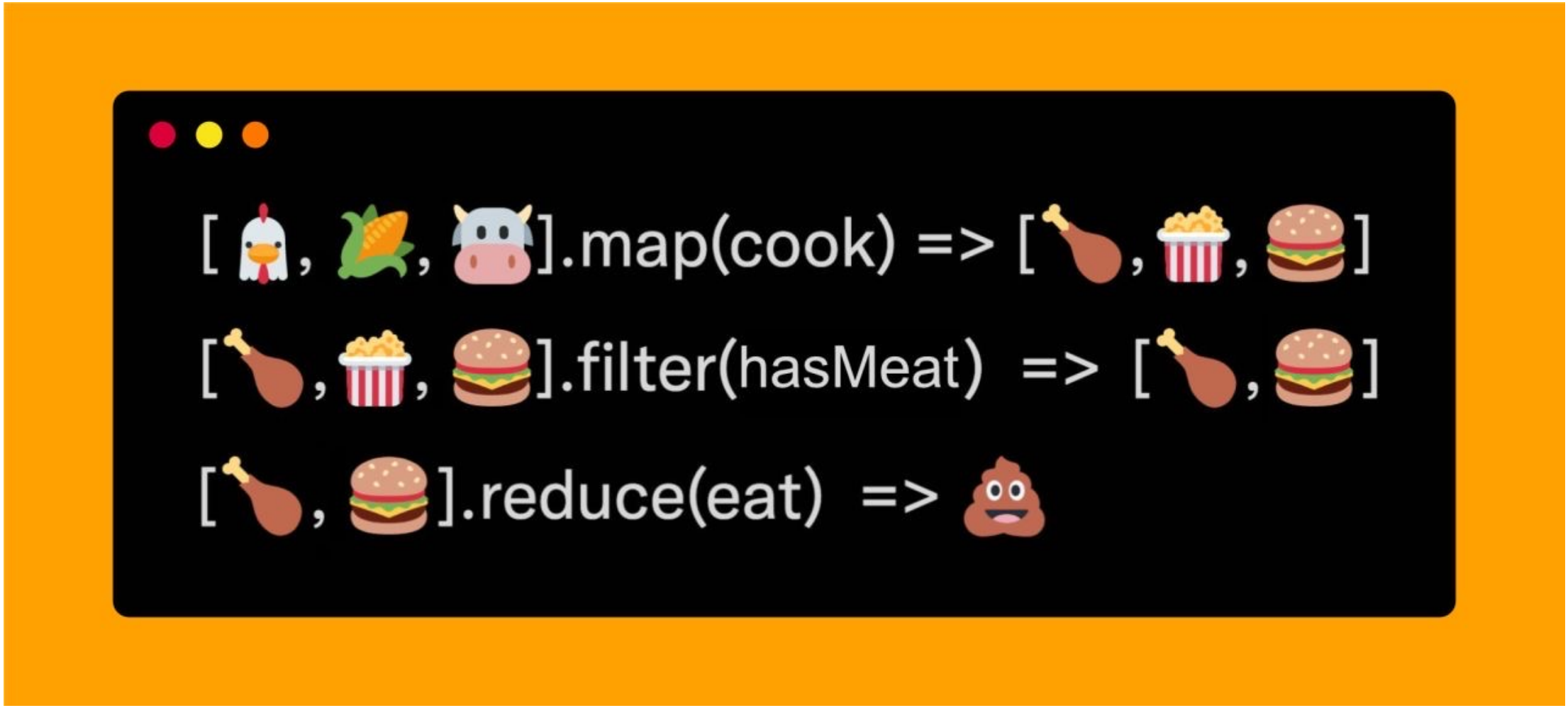
- Function Expression

```
const square = function(x) {
  return x * x
}
```

- Arrow Function

```
const square = (x) => {
  return x * x
}
```


function makeCounter() {
let count = 0;
return function() {
return count++;
};
}
let counter1 = makeCounter();
let counter2 = makeCounter();
console.log(counter1()); // 0
console.log(counter1()); // 1
// Are counter1 and counter2 independent?
console.log(counter2()); // ?


[🐔, 🌽, 🐮].map(cook) => [🍗, 🍿, 🍔]
[🍗, 🍿, 🍔].filter(hasMeat) => [🍗, 🍔]
[🍗, 🍔].reduce(eat) => 💩

```
const x = undefined
if(x === x){
  console.log('True')
} else {
  console.log('False')
}
```

```
const x = 1 * undefined
if(x === x){
  console.log('True')
} else {
  console.log('False')
}
```

A True, True

B False, True

C True, False

D False, False

Git Cheat Sheet

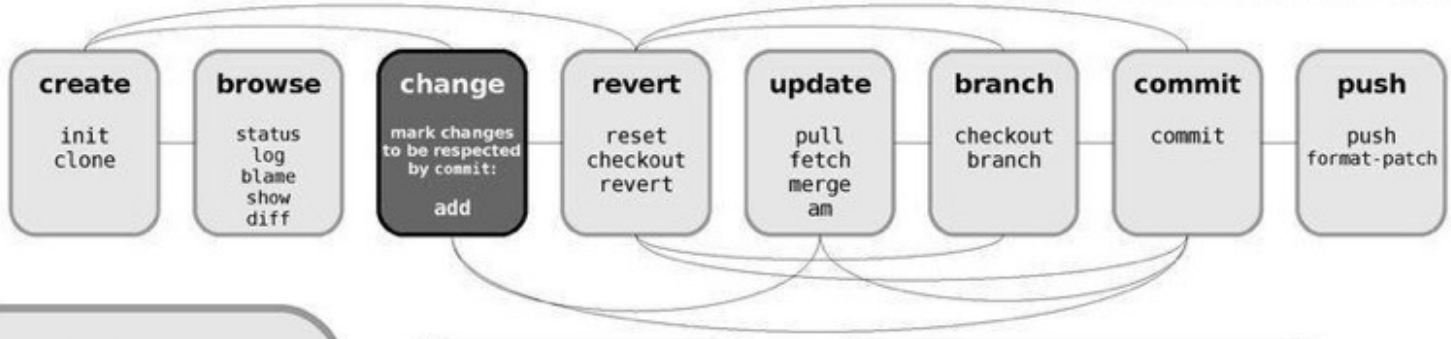
by Jan Krüger <jk@jk.gs>, http://jan-krueger.net/git/
Based on work by Zack Rusin

Basics

Use git help [*command*] if you're stuck.

`master`	default devel branch
`origin`	default upstream branch
`HEAD`	current branch
`HEAD^`	parent of HEAD
`HEAD~4`	great-great grandparent of HEAD
`foo..bar`	from branch *foo* to branch *bar*

(left to right) Command Flow



Create

From existing files

```
git init
git add .
```

From existing repository

```
git clone ~/old ~/new
git clone git://...
git clone ssh://...
```

View

```
git status
git diff [oldid newid]
git log [-p] [file|dir]
git blame file
git show id (meta data + diff)
git show id:file
git branch (shows list, * = current)
git tag -l (shows list)
```

Revert

In Git, `revert` usually describes a new commit that undoes previous commits.

```
git reset --hard (NO UNDO)
        (reset to last commit)
git revert branch
git commit -a --amend
        (replaces prev. commit)
git checkout id file
```

Publish

In Git, `commit` only respects changes that have been marked explicitly with add.

```
git commit [-a]
        (-a: add changed files automatically)
git format-patch origin
        (create set of diffs)
git push remote
        (push to origin or remote)
git tag foo
        (mark current version)
```

Update

```
git fetch (from def. upstream)
git fetch remote
git pull (= fetch & merge)
git am -3 patch.mbox
git apply patch.diff
```

Branch

```
git checkout branch
        (switch working dir to branch)
git merge branch
        (merge into current)
git branch branch
        (branch current)
git checkout -b new other
        (branch new from other and switch to it)
```

Useful Tools

```
git archive
        Create release tarball
git bisect
        Binary search for defects
git cherry-pick
        Take single commit from elsewhere
git fsck
        Check tree
git gc
        Compress metadata (performance)
git rebase
        Forward-port local changes to remote branch
git remote add URL
        Register a new remote repository for this tree
git stash
        Temporarily set aside changes
git tag
        (there's more to it)
gitk
        Tk GUI for Git
```

Conflicts

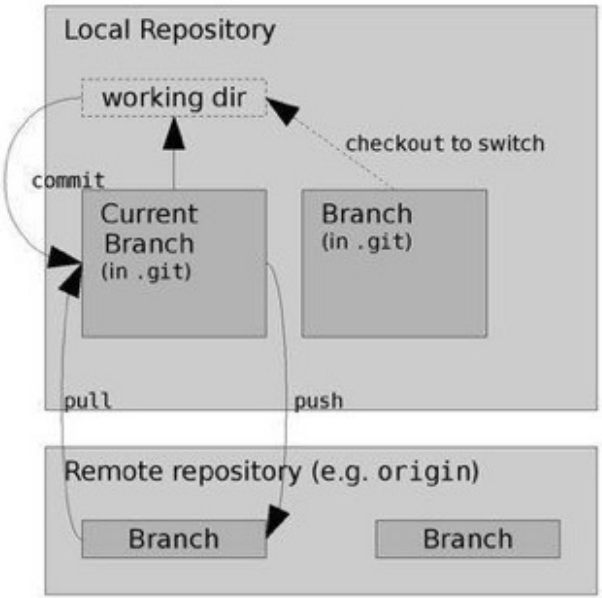
Use add to mark files as resolved.

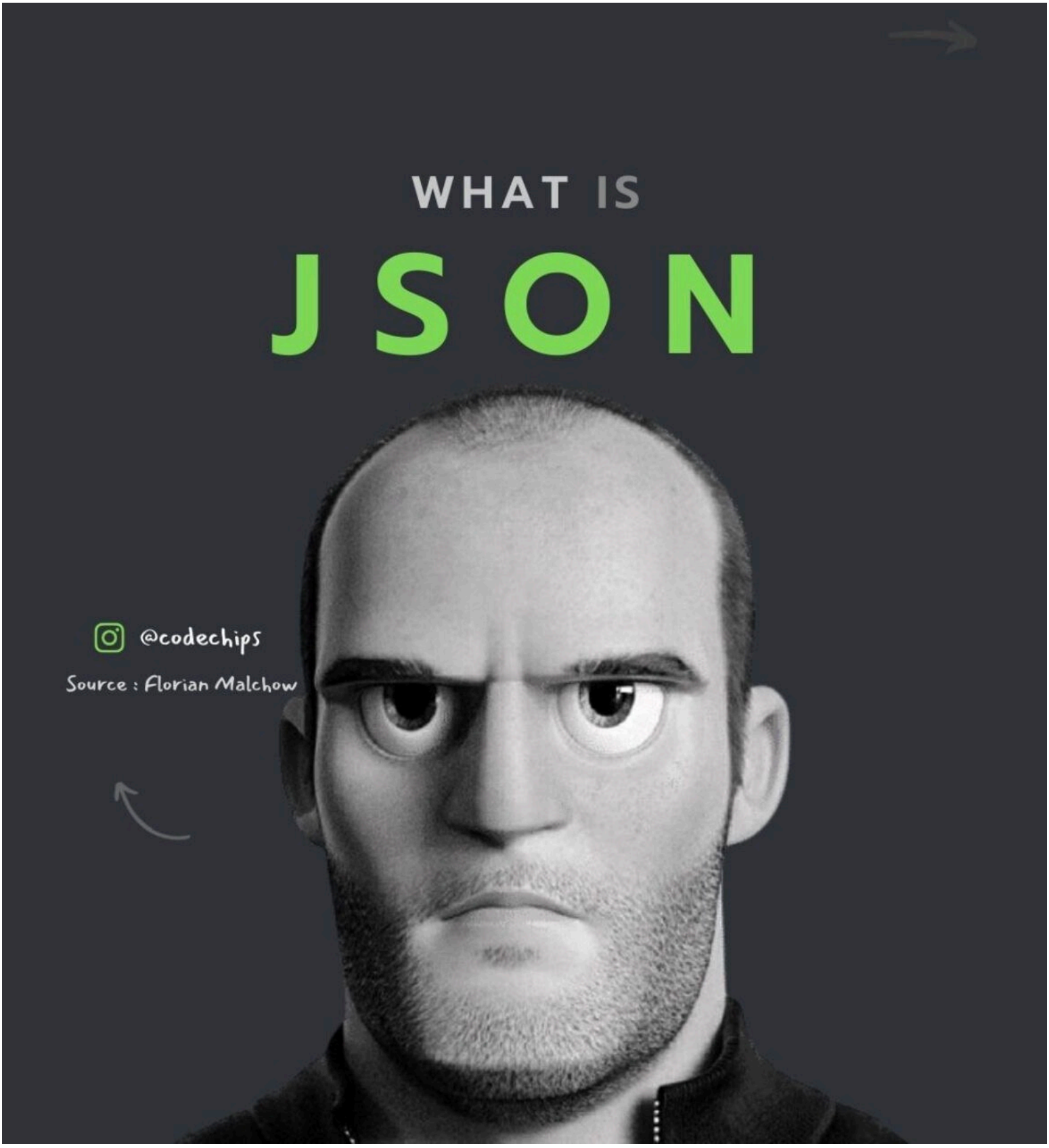
```
git diff [--base]
git diff --ours
git diff --theirs
git log --merge
gitk --merge
```

Tracking Files

```
git add files
git mv old new
git rm files
git rm --cached files
        (stop tracking but keep files in working dir)
```

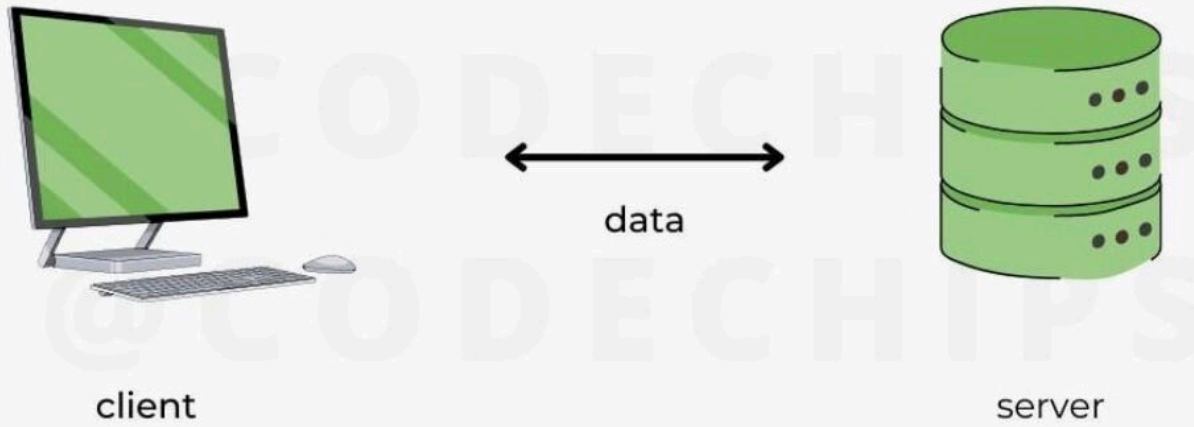
Structure Overview



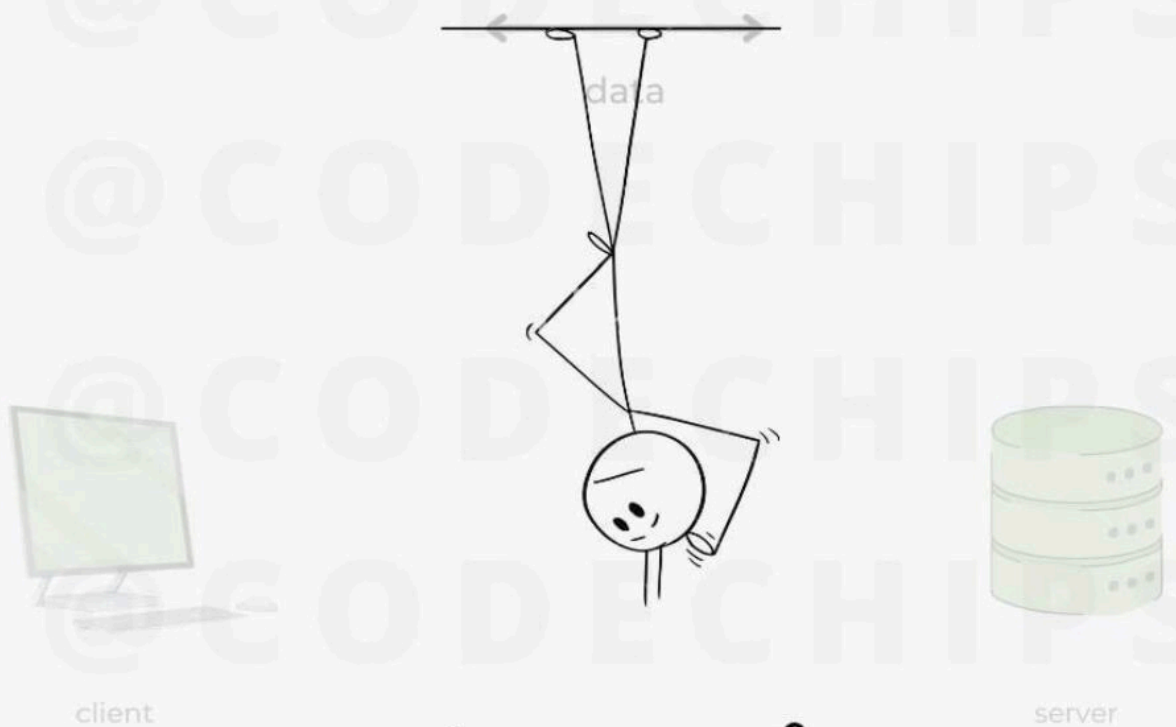

WHAT IS
JSON
@codechips
Source : Florian Malchow

Lets say there is website to display information about pizzas

And they need to talk to
each other and transfer data



But in what format can the data be tranferred ?



What about plain text format ?

The 1st Pizza is **Neapolitan** Pizza which has a **thin** crust and **Mozzarella** cheese, Tomatoes, Basil toppings .

The 2nd Pizza is **New York-Style** Pizza which has a **thick** crust and **Mozzarella** cheese, Tomatoes toppings

It is easy to read for us but not easy to interpret on the client side

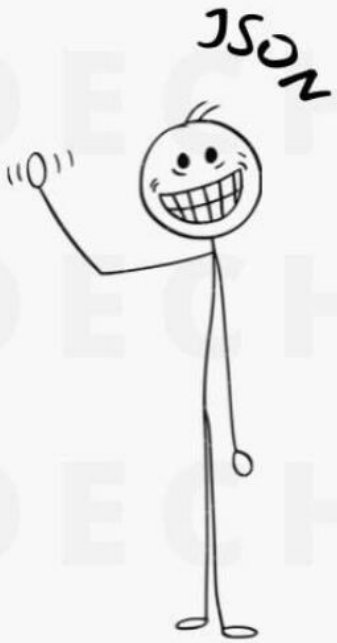
Then XML was introduced

```
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <pizza>
    <id>01</id>
    <name>Neapolitan  Pizza</name>
  </pizza>
  <pizza>
    <id>02</id>
    <name>New York-Style Pizza</name>
  </pizza>
</root>
```

XML has been in use for many years, still the format could be bulky and cause too much overhead for web services

That is when JSON came into play

JSON stands for **J**avaScript **O**bject **N**otation, a lightweight format for **storing** and **transporting** data from a server to a web page, it is "self-describing" and **easy** to understand



JSON Syntax Rules

```
{
   "name" : "value" ,
   "array" : [   ]
}
```

- Data is in name/value (any type) pairs
- Data is separated by commas
- Curly braces hold objects
- Square brackets hold arrays

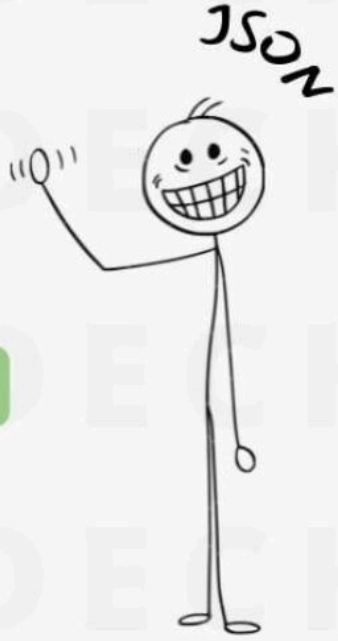
JSON looks like this

```
{
  "pizza": [

    {
      "id":"01",
      "name": "Neapolitan Pizza",
      "toppings": ["Mozzarella","Tomatoes",Basil"]
    },

    {
      "id":"02",
      "name": "New York-Style Pizza",
      "toppings": ["Mozzarella","Tomatoes","Basil"]
    }
  ]
}
```

Why is JSON preferred ?

Less Verbose

Faster



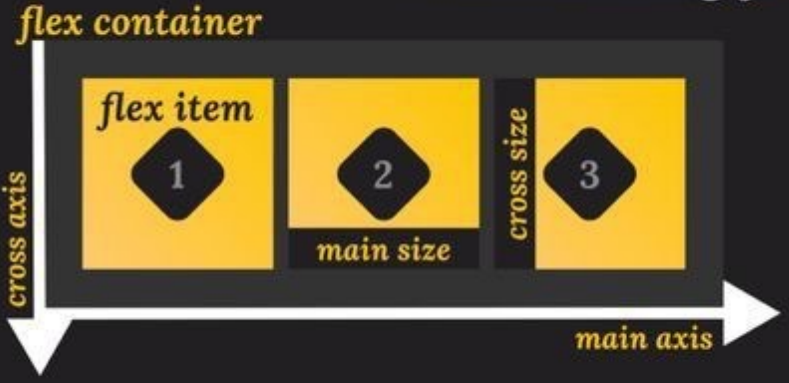
Objects Align in Code

Readable

Structure Matches the Data

Flexbox cheatsheet

terminology



flex container

flex-direction

Sets how the flex items are placed inside the container

flex-wrap

Tells the container whether to wrap the items or not

justify-content

Tells the container how to align the items horizontally

*can also be set to space-around | space-evenly

align-content

Tells the container how to align the items vertically

*can also be set to space-between | space-around | space-evenly

flex items

Interactive examples:

flex-basis

Sets the main size of a flex item

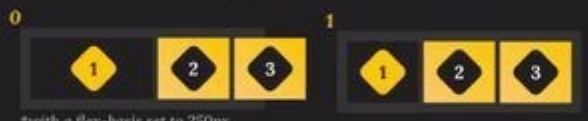
flex-grow

Sets how much space should be taken up by the flex item

*with a flex-grow set to 0 for the other items

flex-shrink

Sets how much should a flex item shrink relative to others



*with a flex-basis set to 250px

align-self

Sets how individual items should be aligned inside the container

order

Sets the ordering of the flex items

CONTAINER

display -

Establishes a new grid formatting context for children

display: **grid**

display:
inline-grid

display: **subgrid**

grid-template -

Defines the rows and columns of the grid

columns: **12px 12px 12px;**
rows: **12px 12px 12px;**

columns: **repeat(3, 12px);**
rows: **repeat(3, auto);**

columns: **8px auto 8px;**
rows: **8px auto 12px;**

columns: **22% 22% auto;**
rows: **22% auto 22%;**

CONTAINER

align-items -

Aligns content in a grid item along the column axis

align-items: **start**

align-items: **end**

align-items: **center**

align-items: **stretch**

justify-content -

Justifies all grid content on row axis when total grid size is smaller than container

start

end

center

stretch

space-around

space-between

space-evenly

CONTAINER

grid-gap -

Specifies the size of column and row gutters

grid-row-gap: **1px;**
grid-column-gap: **9px;**

grid-gap: **1px 9px;**

grid-gap: **6px;**

justify-items -

Aligns content in a grid item along the row axis

justify-items:
start

justify-items:
end

justify-items:
center

justify-items:
stretch

CHILDREN

grid-column -

Determines an items column-based location within the grid

start: 1;
end: 3;

start: 2;
end: 4;

start: span 3;

grid-column:
2 / 3;

grid-column: 2 / span 2;

grid-row -

Determines an items row-based location within the grid

start: 1;
end: 3;

start: span 3;

start: 2;
end: 4;

grid-row:
1 / 3;

grid-row: 1 / span 3;

CHILDREN

justify-self -

Aligns content for a specific grid item along the row axis

justify-self:
start;

justify-self:
end;

justify-self:
center;

justify-self:
stretch;

align-self -

Aligns content for a specific grid item along the column axis

align-self:
start;

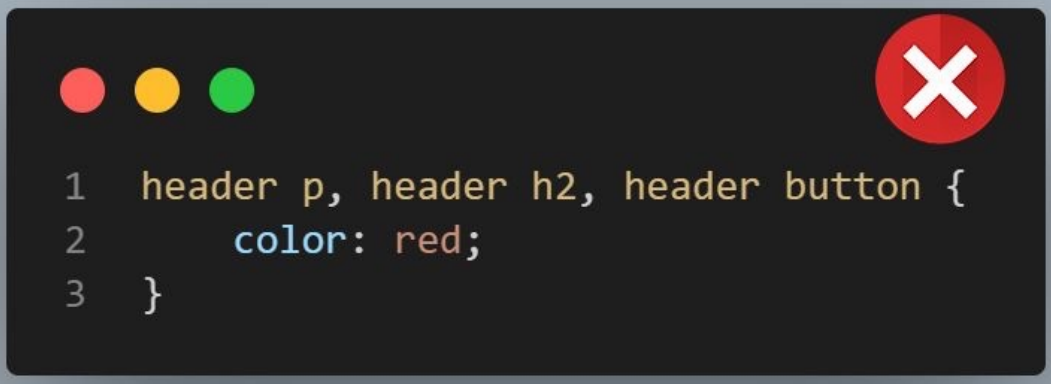
align-self:
end;

align-self:
center;

align-self:
stretch;

CSS Tips:

:is() selector



```
1  header p, header h2, header button {
2      color: red;
3  }
```

Instead of using long selectors like in the example above we can use :is() selector to group things and get much shorter and more readable code

```
1  header :is(p, h2, button) {
2      color: red;
3  }
```

This also works with grouping parents

```
1  :is(header, footer, section) p {
2      color: red;
3  }
```

1 type="text"

I am a Text Input

2 type="password"

●●●●

3 type="search"

Search me

4 type="url"

https://google.com

5 type="email"

hey@example.com

6 type="tel"

+919876543210

7 type="number"

100

8 type="button"

Tweet

9 type="submit"

Submit

10 type="reset"

Reset

11 type="file"

Choose File No File Chosen

12 type="checkbox"

13 type="radio"

14 type="range"

15 type="datetime-local"

dd/mm/yyyy, --:-- --

16 type="month"

--------- ----

17 type="week"

Week --, ----

18 type="time"

--:-- --

19 type="date"

dd/mm/yyyy

20 type="hidden"

"input" elements of type "hidden" let web developers include data that cannot be seen or modified by users when a form is submitted.

21 type="color"

22 type="image"

HTML Input Elements

CSS Units CheatSheet

px - Absolute value in pixel

em - Relative to font-size of element in the case of width and height

rem - Relative to font-size of the root element

% - Relative to the value of parent element

vh - 1% of viewport height

vw - 1% of viewport width

vmin - Relative to viewport's smaller dimension

1vmin = min(1vw, 1vh)

vmax - Relative to viewport's larger dimension

1vmax = max(1vw, 1vh)

by Nishant

Detect if the user's browser is mobile

```
if(/Android|webOS|iPhone|OperaMini/i.test(navigator.userAgent)){
   // true for mobile device
   console.log("mobile")
  }
else{
 // false for not mobile device
 console.log("not mobile device");
}
```

 @elias_kibret2

 Elias Kibret

 @EliasKibret2

rootlearn.com

JWT

Web

JSON

Tokens

It's a popular token based authentication standard

A token consists of 3 parts separated by dots

HHHHHH . PPPPPP . SSSSSSSSS

Header

Includes metadata info & contains two parts

Token type

```
{
  "typ": "JWT",
  "alg": "HS256"
}
```

Hashing algorithm

Payload

Contains statements about an entity and additional data aka "claims"

```
{
  "userid": 'Manish",
  "email": "abc@rootlearn.com",
  "exp": 1427321171,
  "sub": "78954"
}
```

3 types of claims

- Public
- Registered
- Private

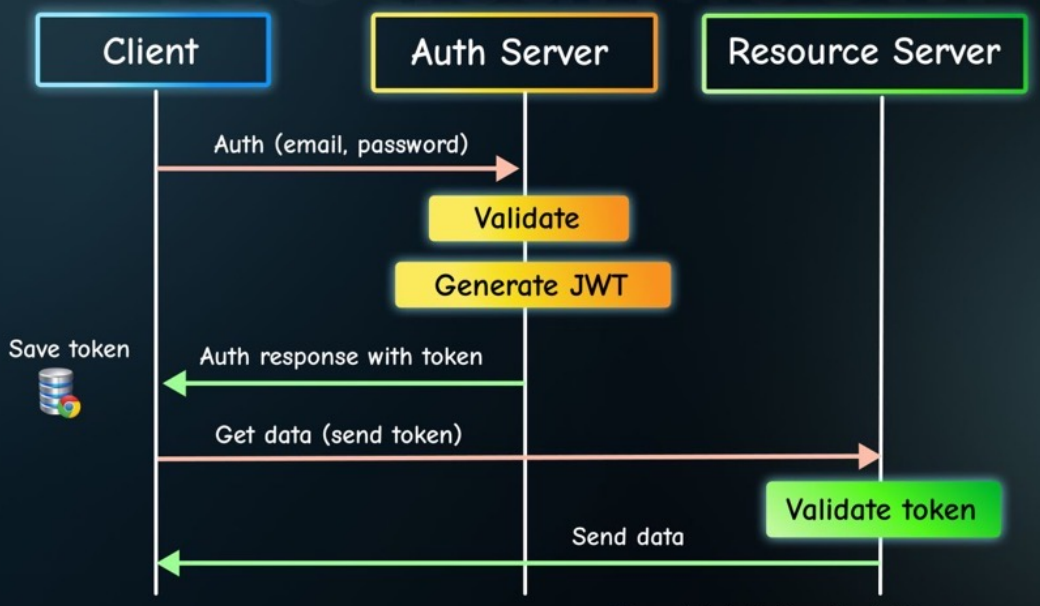
Signature

The signature is made up with the help of 4 parts

- Base64 header
- Base64 payload
- Secret
- Cryptographic algo

* All of these are Base64Url encoded to form the 3 parts in a JWT

Basic JWT Auth Flow

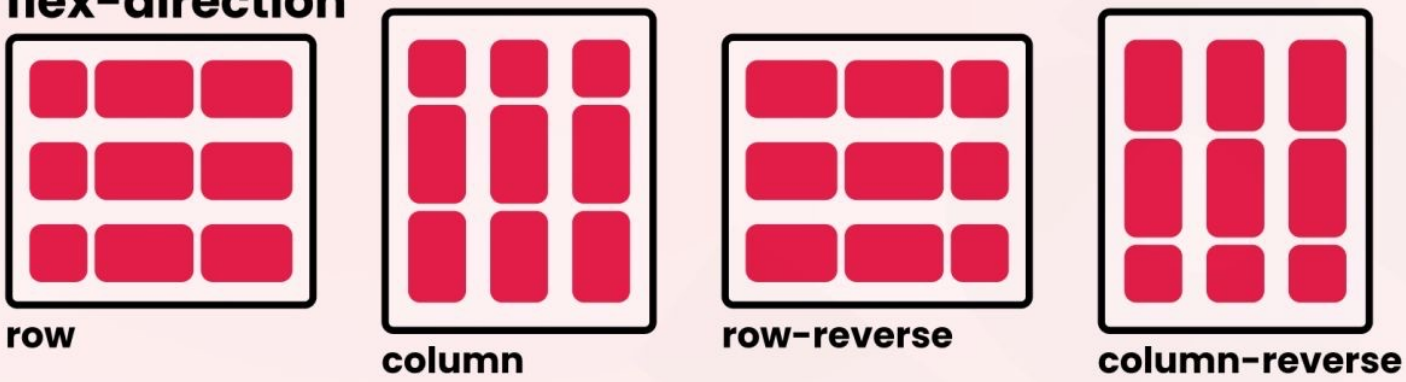


*Doesn't include the refresh token flow

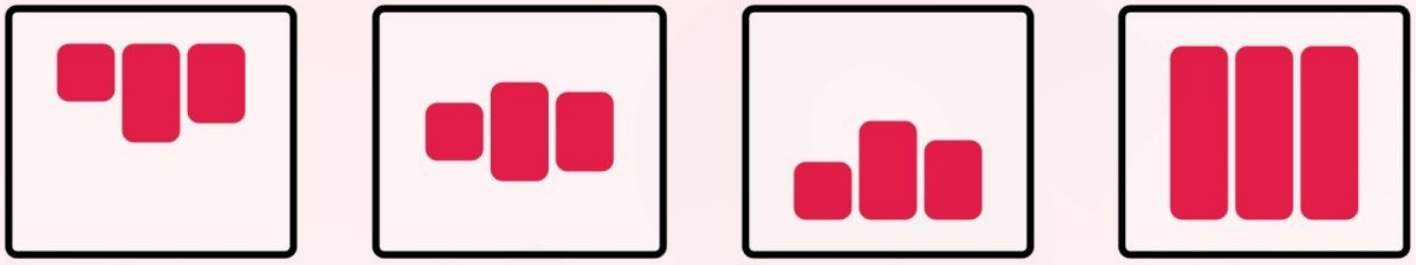
Manish Poduval

CSS Flexbox

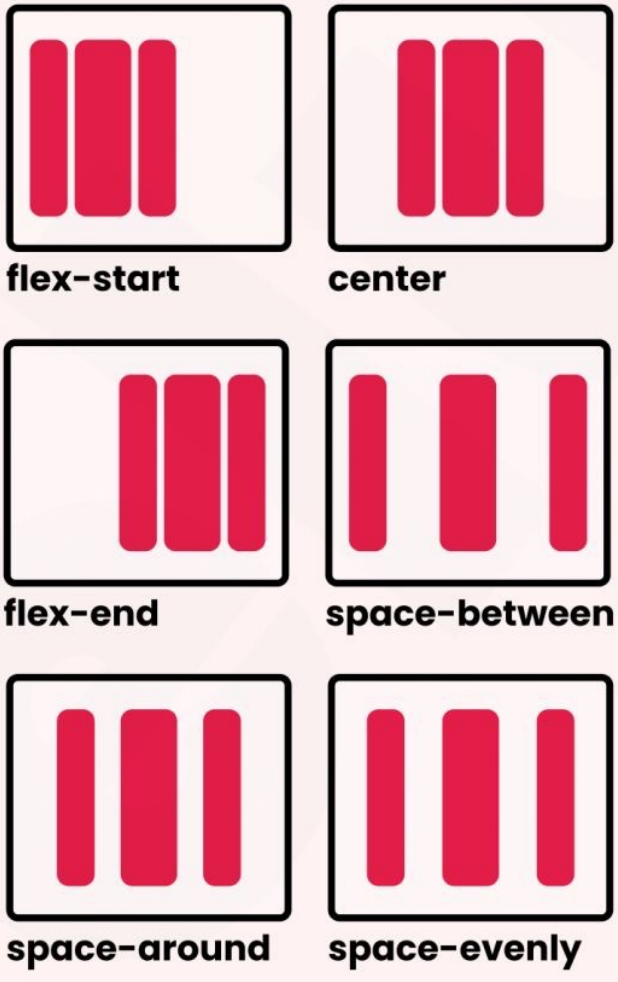
flex-direction



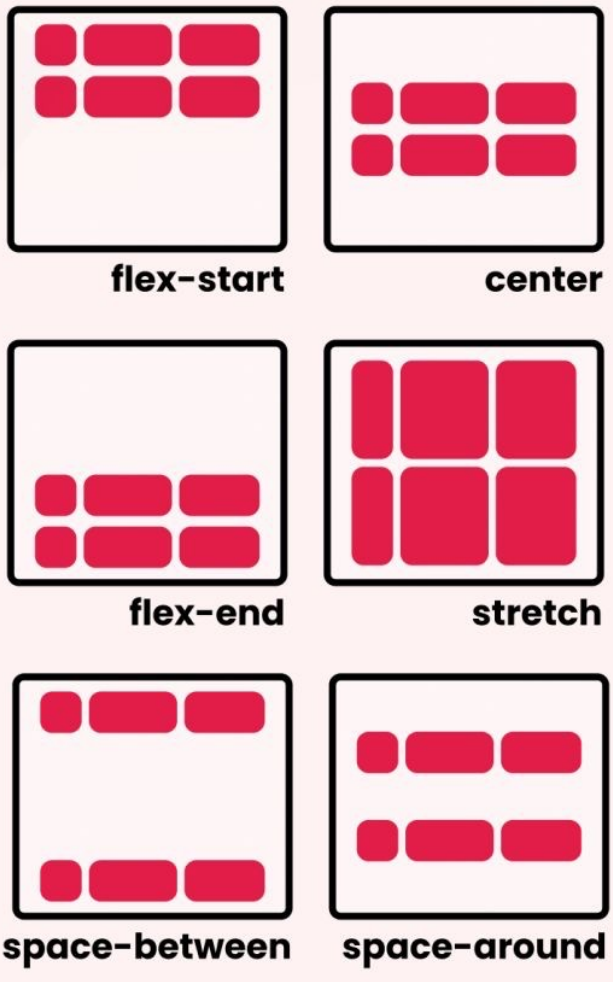
align-items



justify-content



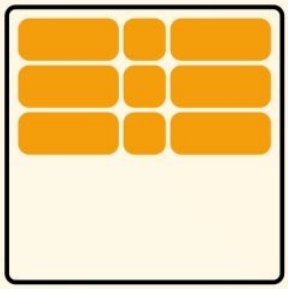
align-content



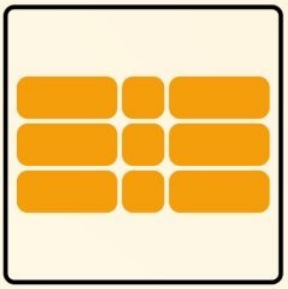
@eludadev

CSS Grid Layout

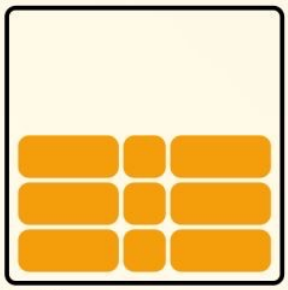
align-content



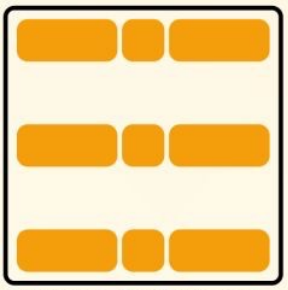
start



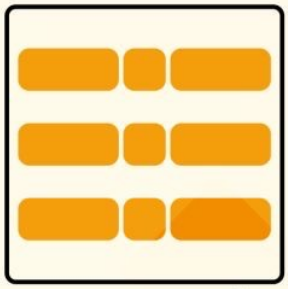
center



end



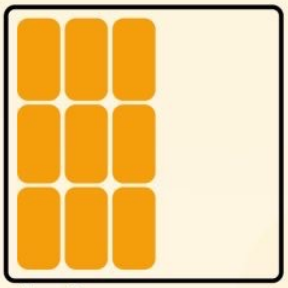
space-between



space-around

stretch

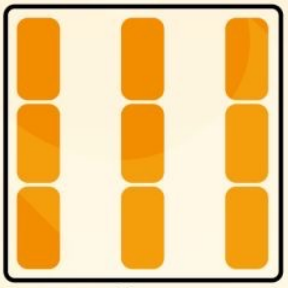
justify-content



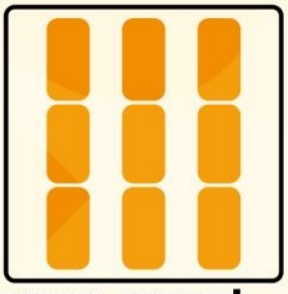
start

center

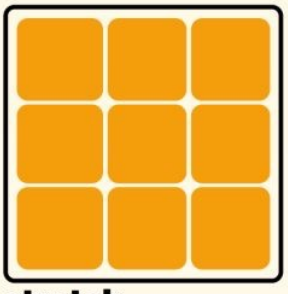
end



space-between

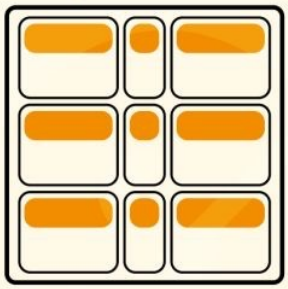


space-around

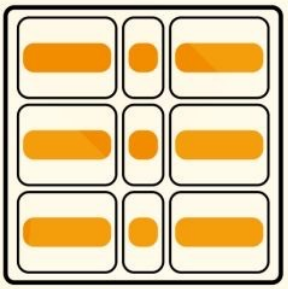


stretch

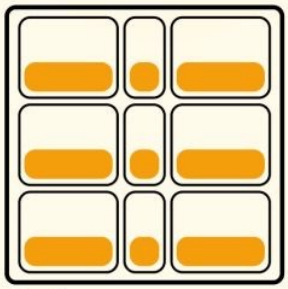
align-items



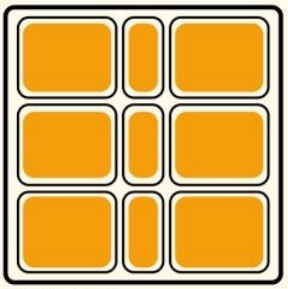
start



center

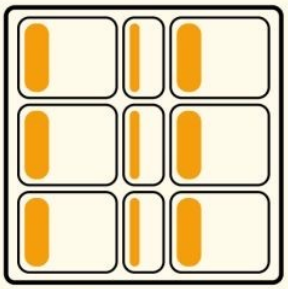


end

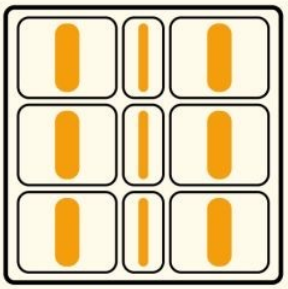


stretch

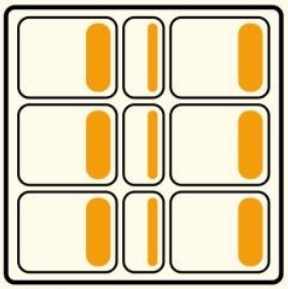
justify-items



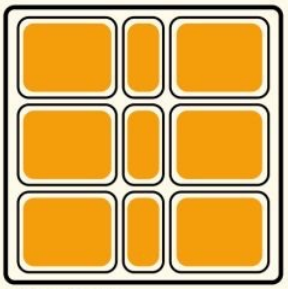
start



center

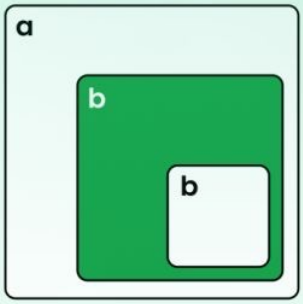


end



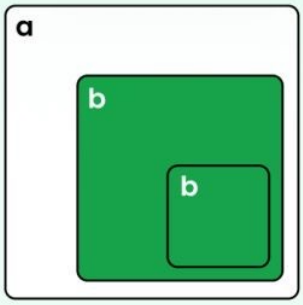
stretch

CSS Selectors



a > b Child Combinator

Select all *b* elements that are directly inside of *a* elements.



a b Descendent Combinator

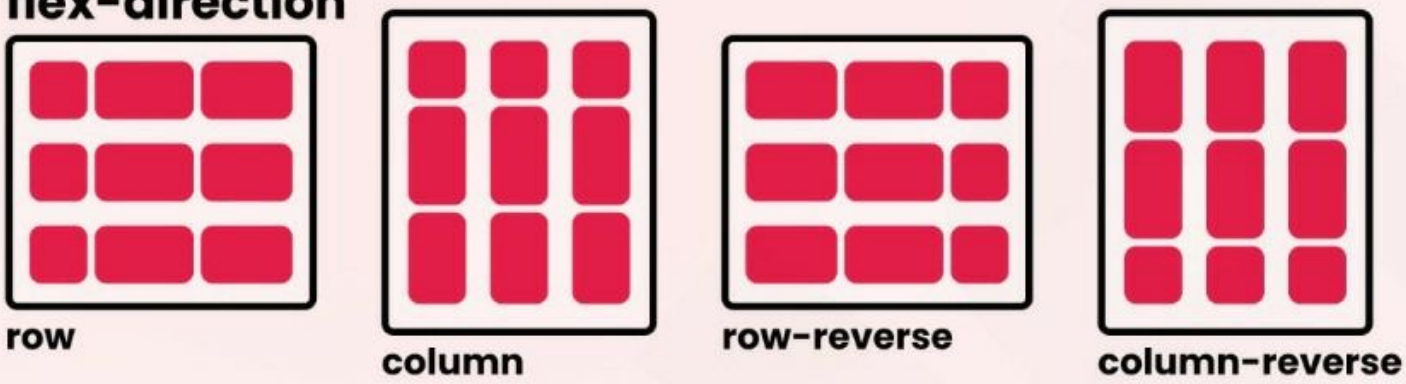
Select all *b* elements that are anywhere inside of *a* elements.

	**a + b** Adjacent sibling combinator	Select all *b* elements that are immediately next to *a* elements.
	**a ~ b** General sibling combinator	Select all *b* elements that are anywhere after *a* elements.
	**.cl** Class selector	Select all elements that have the *cl* class name.
	**a.cl** Tag + Class selector	Select all *a* elements that have the *cl* class name.
	**.cl1.cl2** Multiclass selector	Select all elements that have both the *cl1* and *cl2* class names.
	**a[x=y]** Attribute selector	Select all *a* elements that have the *x* attribute set to *y*.
	**#id1** ID selector	Select the element with the *id1* ID name.
	* Universal selector	Select all elements.

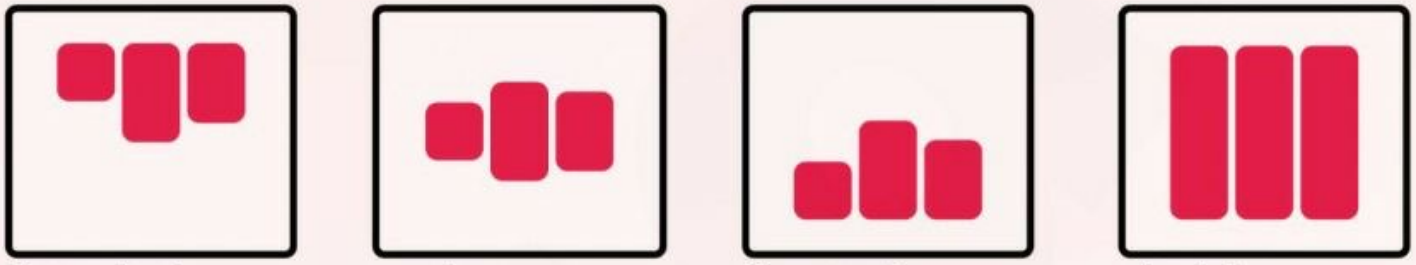
@eludadev

CSS Flexbox

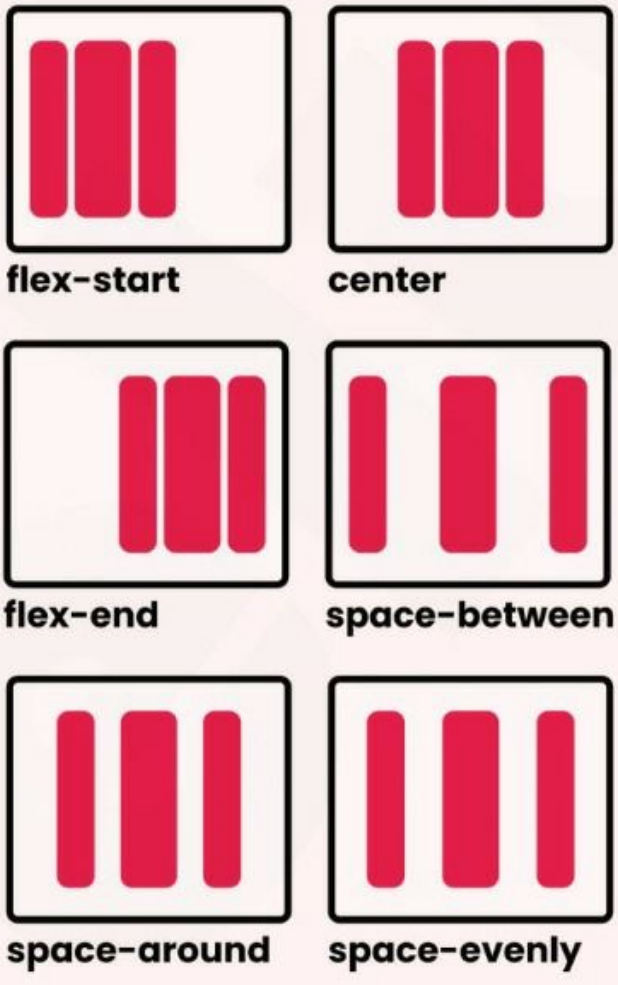
flex-direction



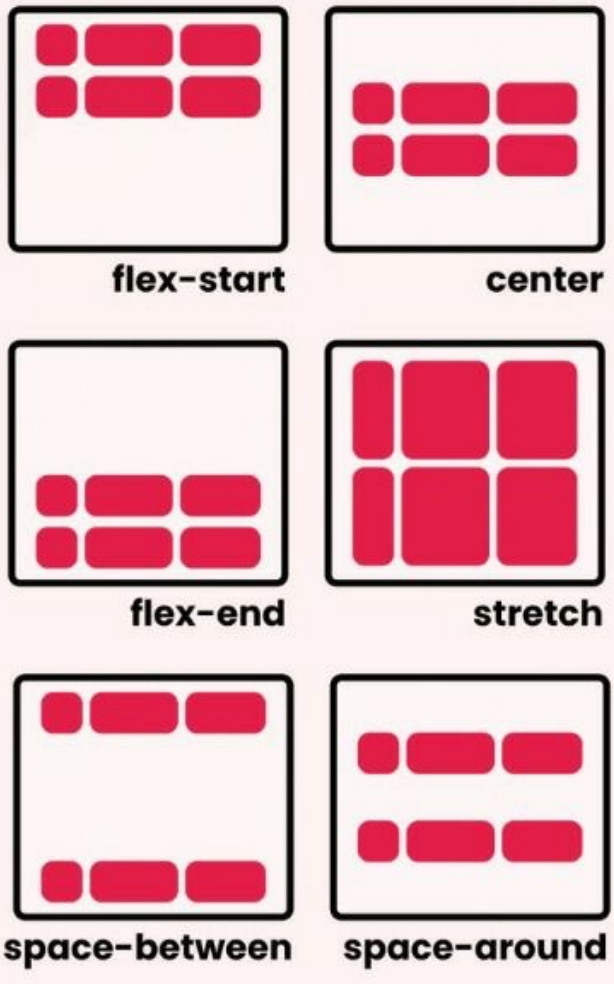
align-items



justify-content

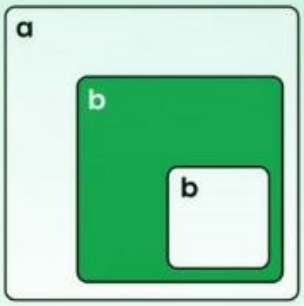


align-content



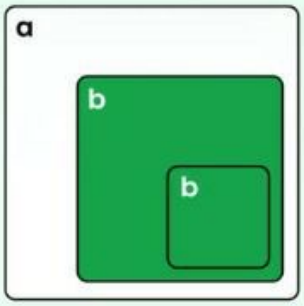
@eludadev

CSS Selectors



a > b Child Combinator

Select all *b* elements that are directly inside of *a* elements.

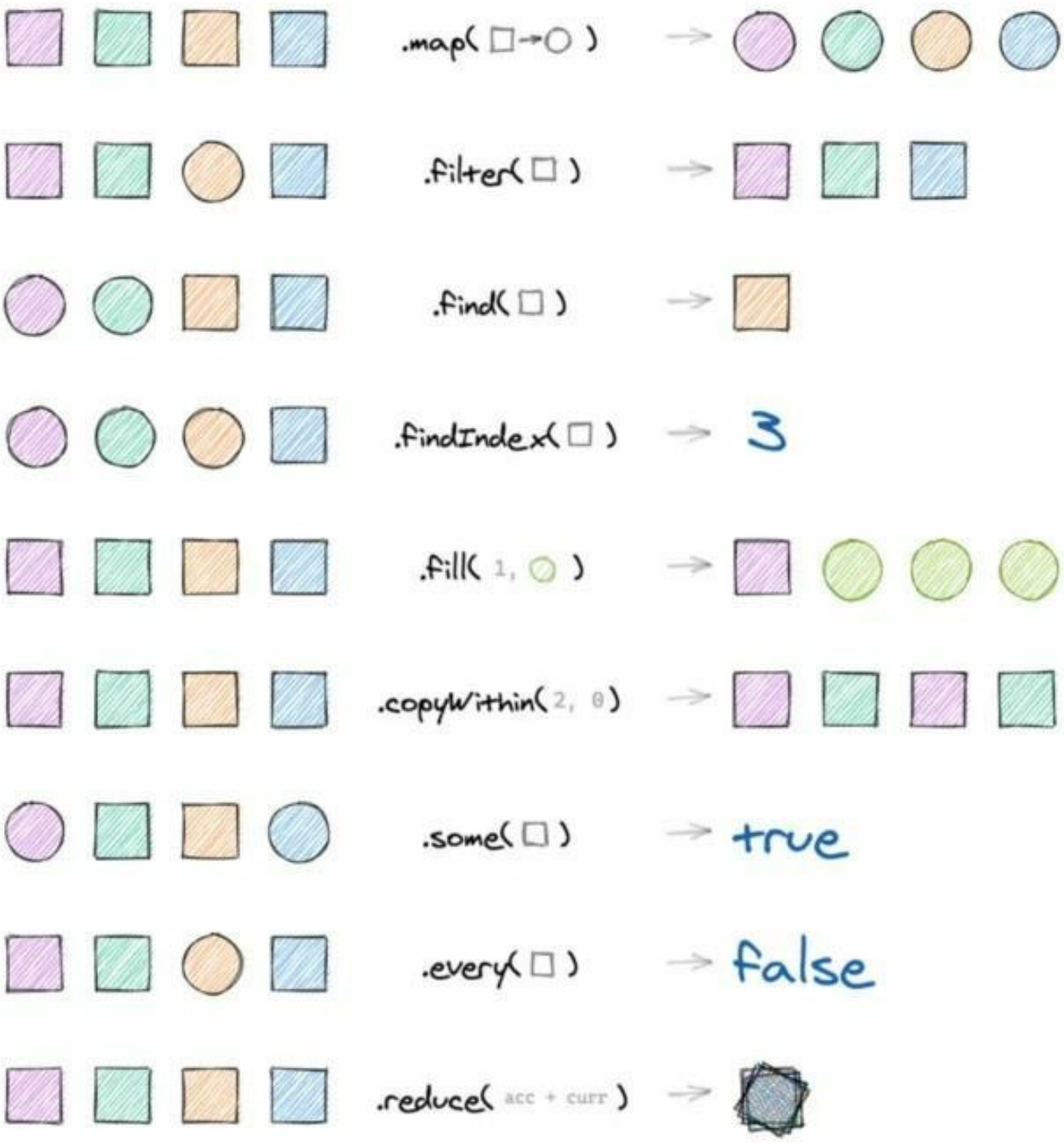


a b Descendent Combinator

Select all *b* elements that are anywhere inside of *a* elements.

	**a + b**	**Adjacent sibling combinator**	Select all *b* elements that are immediately next to *a* elements.
	**a ~ b**	**General sibling combinator**	Select all *b* elements that are anywhere after *a* elements.
	**.cl**	**Class selector**	Select all elements that have the *cl* class name.
	**a.cl**	**Tag + Class selector**	Select all *a* elements that have the *cl* class name.
	**.cl1.cl2**	**Multiclass selector**	Select all elements that have both the *cl1* and *cl2* class names.
	**a[x=y]**	**Attribute selector**	Select all *a* elements that have the *x* attribute set to *y*.
	**#id1**	**ID selector**	Select the element with the *id1* ID name.
	*****	**Universal selector**	Select all elements.

@eludadev


.map(□→○)
.filter(□)
.find(□)
.findIndex(□)
3
.fill(1, ○)
.copyWithin(2, 0)
.some(□)
true
.every(□)
false
.reduce(acc + curr)

Selector `:nth-child` in #CSS:

:where() in CSS

The :where() CSS pseudo-class function takes a selector list as its arguement, and selects any element that can be selected by one of the selectors in that list.

style.css

```
.card h1, .card h2, .card h3 {
  color: #fafafa
}
```

Applying same style to multiple elements

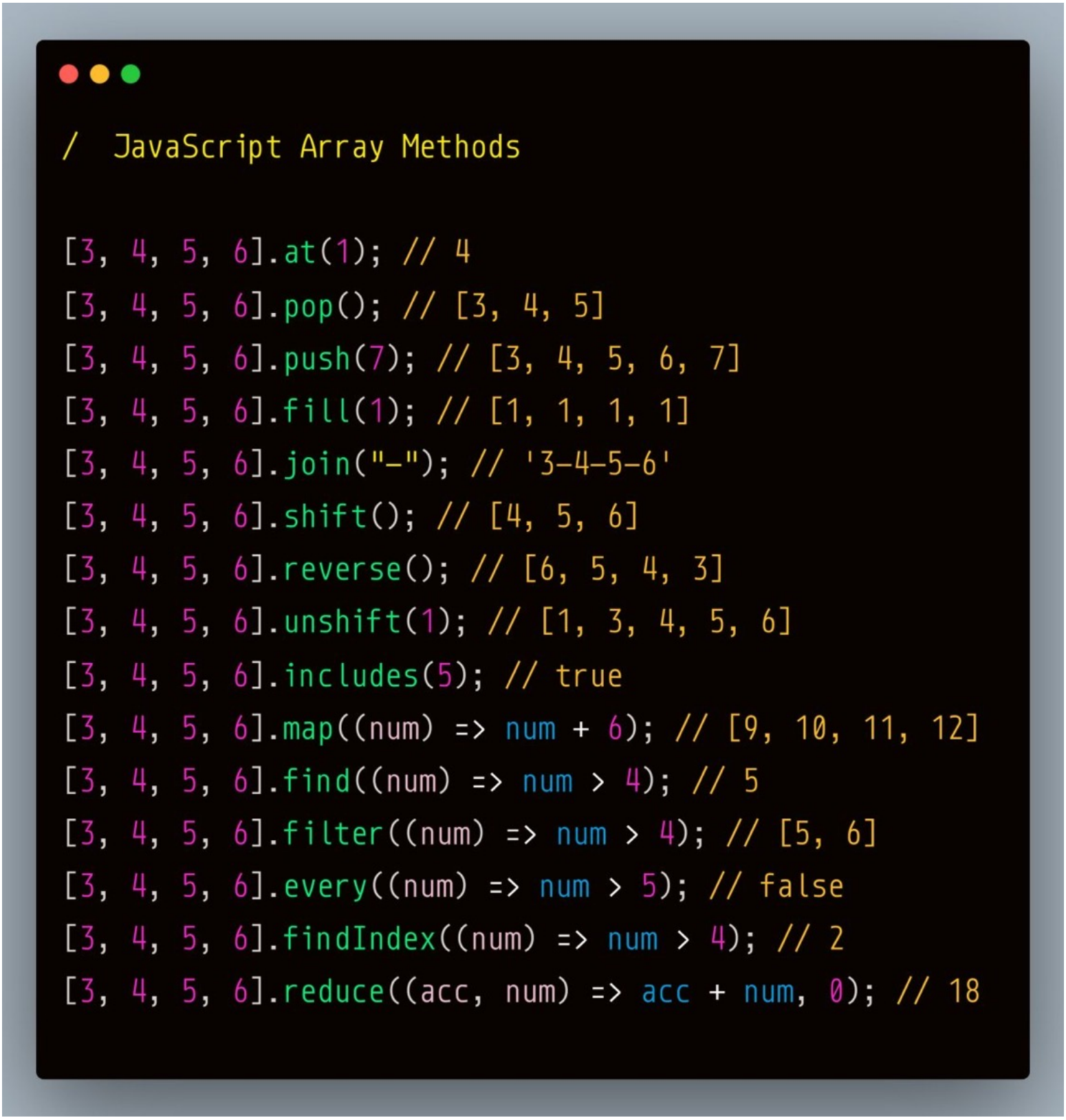
style.css

```
.card :where(h1, h2, h3) {
  color: #fafafa;
}
```

Both will result in same output!

The :where() function accepts a selector list as its arguement and will select elements that can be selected by any rule in the selector list.

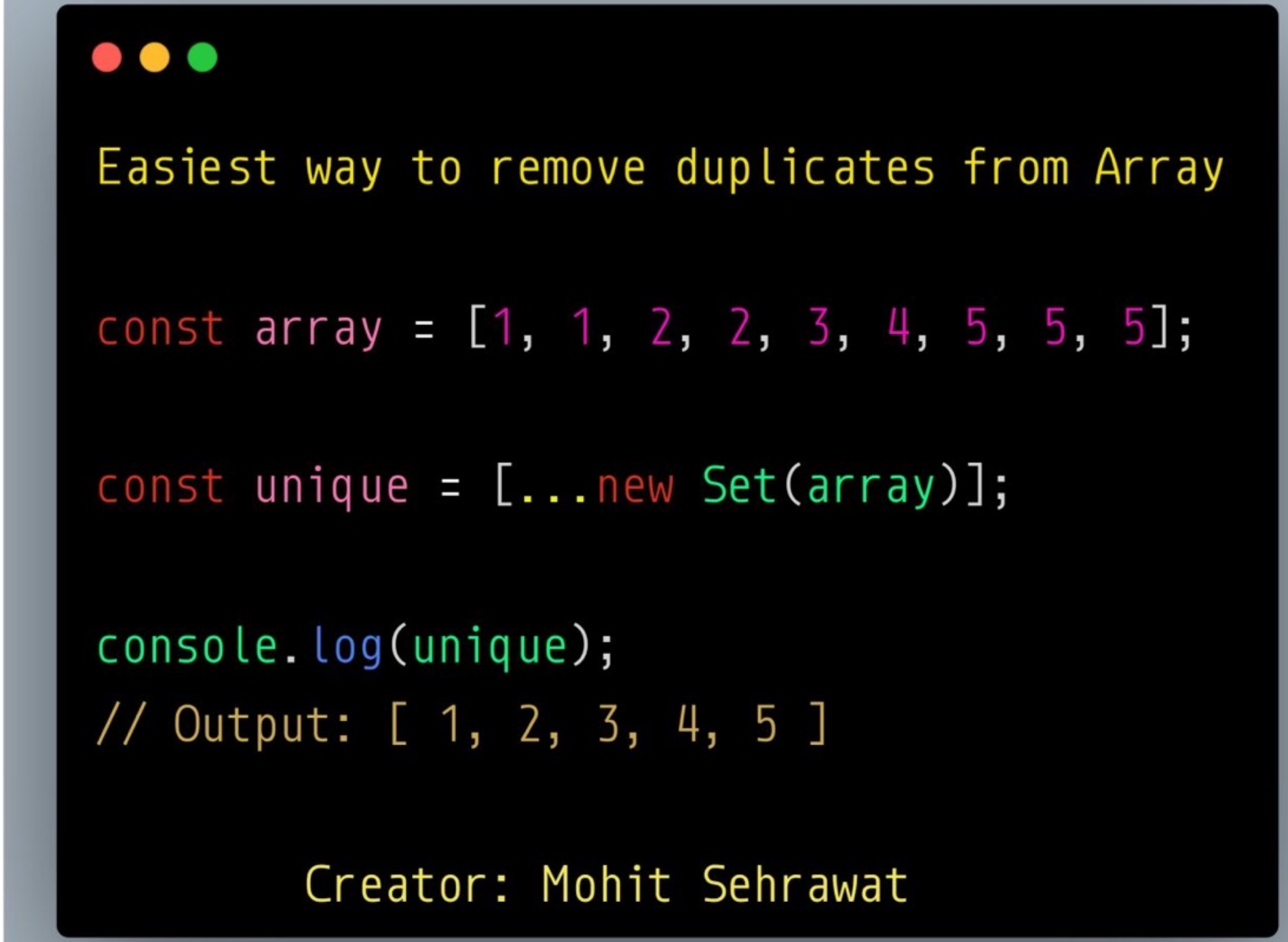
Code with Sloba

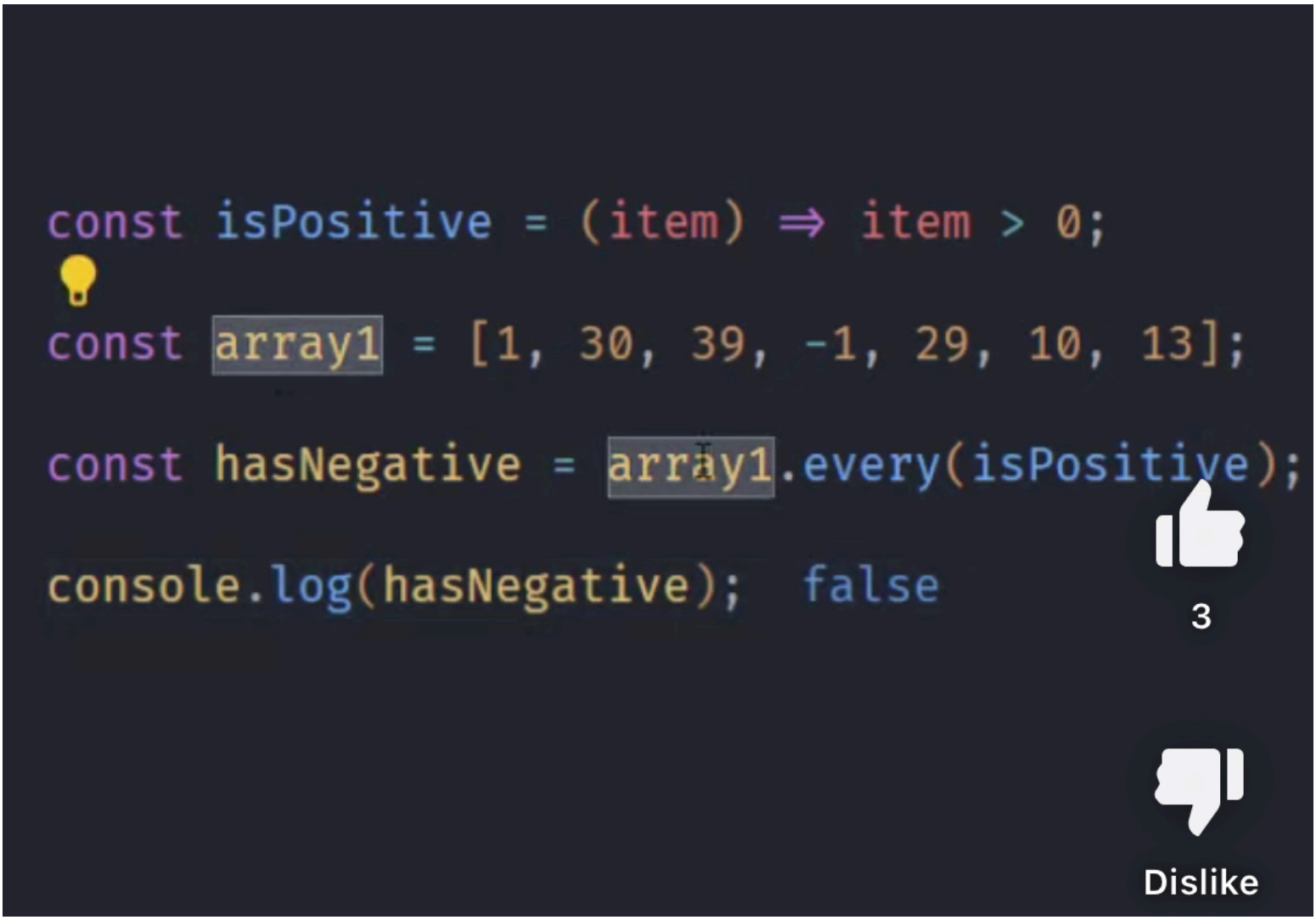

/ JavaScript Array Methods
[3, 4, 5, 6].at(1); // 4
[3, 4, 5, 6].pop(); // [3, 4, 5]
[3, 4, 5, 6].push(7); // [3, 4, 5, 6, 7]
[3, 4, 5, 6].fill(1); // [1, 1, 1, 1]
[3, 4, 5, 6].join("-"); // '3-4-5-6'
[3, 4, 5, 6].shift(); // [4, 5, 6]
[3, 4, 5, 6].reverse(); // [6, 5, 4, 3]
[3, 4, 5, 6].unshift(1); // [1, 3, 4, 5, 6]
[3, 4, 5, 6].includes(5); // true
[3, 4, 5, 6].map((num) => num + 6); // [9, 10, 11, 12]
[3, 4, 5, 6].find((num) => num > 4); // 5
[3, 4, 5, 6].filter((num) => num > 4); // [5, 6]
[3, 4, 5, 6].every((num) => num > 5); // false
[3, 4, 5, 6].findIndex((num) => num > 4); // 2
[3, 4, 5, 6].reduce((acc, num) => acc + num, 0); // 18

JavaScript String Methods Cheetsheet

```
"JavaScript".slice(0, 4); // "Java"
"JavaScript".substring(0, 4); // "Java"
"JavaScript".replace("Script", "Hello"); // "JavaHello"
"JavaScript".toUpperCase(); // "JAVASCRIPT"
"JAVASCRIPT".toLowerCase(); // "javascript"
"JavaScript".charAt(1); // "a"
"JavaScript".charCodeAt(1); // 97
"JavaScript".split(""); // [J, a, v, a, S, c, r, i, p, t]
"JavaScript".indexOf("a"); // 1
"JavaScript".lastIndexOf("a"); // 3
"JavaScript".startsWith("J"); // true
"JavaScript".endsWith("t"); // true
"JavaScript".includes("Java"); // true
"JavaScript".concat(" is cool"); // "JavaScript is cool"
" JavaScript ".trim(); // "JavaScript"
" JavaScript ".trimStart(); // "JavaScript "
" JavaScript ".trimEnd(); // " JavaScript"
```

Creator: Mohit Sehrawat


Easiest way to remove duplicates from Array
const array = [1, 1, 2, 2, 3, 4, 5, 5, 5];
const unique = [...new Set(array)];
console.log(unique);
// Output: [1, 2, 3, 4, 5]
Creator: Mohit Sehrawat


const isPositive = (item) ⇒ item > 0;
const array1 = [1, 30, 39, -1, 29, 10, 13];
const hasNegative = array1.every(isPositive);
console.log(hasNegative); false
3
Dislike

CSS Units - Cheatsheet

CSS Units determines the size of a property you're setting for an element of its content

1. Absolute Units

Units that are "absolute" are the same size regardless of the parent element or window size .

Absolute Units		Description
px	→	1px = 1/96th of 1 inch (96px = 1 inch)
pt	→	1pt = 1/72th of 1 inch
mm	→	1mm = 1/10th of 1cm
cm	→	Centimeters
pc	→	12pt=1pc
in	→	Inches

2. Relative Units

Relative units are useful for styling responsive sites because they scale relative to the parent or window size (depending upon the unit)

Relative Units		Description
%	→	Relative to the parent element
em	→	Relative to the font size of the element
rem	→	Relative to the font size of the root element
ch	→	Relative to the width of the "0" (zero)
vh	→	Relative to 1% of the viewport's height.
vw	→	Relative to 1% of the viewport's width.
vmin	→	Relative to 1% of the viewport's smaller dimension.
vmax	→	Relative to 1% of the viewport's larger dimension.

Pradeep Pandey

JavaScript Higher Order Functions Cheetsheet

```
[1, 2, 3, 4].filter((item) => item % 2); // [2, 4]
[1, 2, 3, 4].map((item) => item * 2); // [2, 4, 6, 8]
[1, 2, 3, 4].reduce((acc, curr) => acc + curr, 0); // 10
[1, 2, 3, 4].some((item) => item > 3); // true
[1, 2, 3, 4].every((item) => item > 2); // false
[1, 2, 3, 4].find((item) => item > 2); // 3
[1, 2, 3, 4].findIndex((item) => item > 2); // 2
[3, 1, 4, 2].sort((a, b) =>  a - b); // [1, 2, 3, 4]
[1, 2, 3, 4].forEach((item) => { console.log(item) });
//Alternative of for loop
```

Creator: Mohit Sehrawat

REDUCE METHOD IN JAVSCRIPT

```
const prices = [8, 41, 17, 29, 33]

const sum = prices.reduce((totalPrice, currentPrice) => {
    return totalPrice += currentPrice
}, 0)
console.log(sum)
Output: 128
```

in Nageshwar Nag

#JavaScript-with-JC

```
1  let name = "Jayesh";
2  function printName() {
3    if (name === "Jayesh") {
4      let name = "JC";
5      console.log(name);
6    }
7    console.log(name);
8  }
9  printName();
10
11 👍A) Jayesh        💡B) Jayesh, JC
12 💖C) JC, JC        😃D) JC, Jayesh
13
```

Jayesh Choudhary
@JayeshMERN

Useful Git Commands Cheatsheet

✅ git init	Initializes a new Git repository
✅ git add <files>	Adds files to the staging area.
✅ git status	Used to check the state of the staging area, as well as the working directory
✅ git log	Used to view the entire commit history.
✅ git commit -m "message"	Used to commit files (locally) on the repository.
✅ git clone	Used to download existing code from a remote repository.
✅ git branch	Used to list all the local branches on the machine.
✅ git merge <branch-name>	Merges the provided branch with the current working branch.
✅ git branch <branch-name>	Used to create a new branch locally.
✅ git branch -d <branch-name>	Used to delete a branch.
✅ git branch -m <new-name>	Used to rename the current working branch.
✅ git checkout <branch-name>	Used to switch from the current branch to another one.
✅ git push <remote> <branch-name	Used to save all commits to the remote repository.
✅ git checkout -b <branch-name>	Creates a new branch and switches to the new one.
✅ git pull <remote>	Used to pull down all the updates from the remote repository.
✅ git rm <file-name>	Used to remove a file from the working directory.
✅ git stash	Used to temporarily remove uncommitted changes.
✅ git reset	Undo the changes to the local files, and restore to the last commit.
✅ git diff	Displays the difference between files in two commits or between a commit and your current repository.

Pradeep Pandey

2 WAYS TO JOIN ARRANGEMENTS IN JAVASCRIPT

```
const array1 = ["1", "2", "3"];
const array2 = ["4", "5", "6"];

const result = array.concat(array2);

//or

const result = [].concat(array1, array2);
const result1 = [...array1, ...arra2];

// Result
["1", "2", "3", "4", "5", "6"];
```

Pradeep Pandey

JavaScript String Method

```
const str = 'Hello World';

str.length; // 11

str.charAt(4); // o

str.endsWith('d'); // true;

str.includes('Worlds'); // true;

str.indexOf(); // 6

str.repeat(2); // "Hello WorldHello World"

str.replace('world', 'Pradeep'); // "Hello Pradeep"

str.slice(6, 10); // "worl"

str.split(' '); // ["Hello , World"]

str.startsWith('Hello'); // true

str.substring(6, 12); // "World";

str.substr(1, 4); // "ello";

str.toLowerCase(); // "hello world"

str.toUpperCase(); // "HELLO WORLD"

'     Hello World!     '.trim(); // "Hello World"
```

Pradeep Pandey

JS

```
<ul>
  <li>Home</li>
  <li>About</li>
  <li>Contact</li>
</ul>
```

```
<ul>
  {
    ['Home','About','Contact'].map((item)=>(
      <li key={item}>
       {item}
      </li>
    ))
  }
</ul>
                          //Elias kibret
```

The draggable Attribute

The draggable is a global attribute that specifies whether an element is draggable or not.

```
<div draggable="true">
   <p> This is a draggable div. </p>
</div>
```

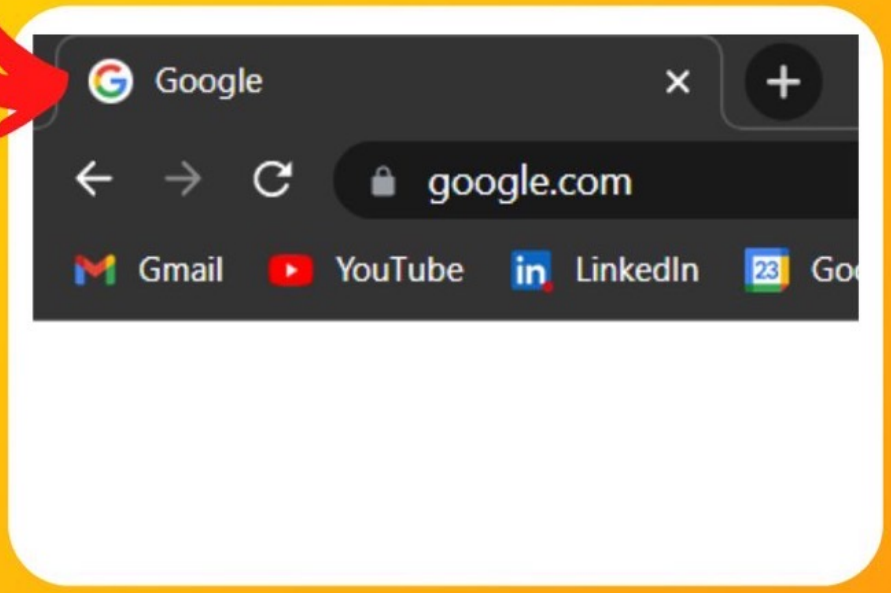
NOTE - To determine the element is dragged or not, a function can be passed to an event listener called ondragstart.

in Sham Gurav

Add Tab Icon in HTML

A favicon (Tab Icon) is a small image displayed next to the page title in the browser tab.

```
<link rel="icon" type="image/x-icon" href="logo.ico">
```



Sham Gurav

Basic Git Commands Cheatsheet

`git config`	⟶	Used to set the email and name of the author
`git init`	⟶	Used to start a new repository.
`git add [file]`	⟶	Adds a file to the staging area.
`git commit -m "message"`	⟶	Used to commit the files
`git status`	⟶	Lists all the files that have to be committed.
`git log`	⟶	Used to view the commit history
`git diff`	⟶	Shows difference between two files or branches
`git branch`	⟶	Lists all the local branches in the repository.
`git branch <branch-name>`	⟶	Creates a new branch.
`git branch -d <branch-name>`	⟶	Used to delete the branch
`git checkout <branch-name>`	⟶	Used to switch from one branch to another
`git push <remote> <branch-name>`	⟶	Used to save commits in the remote repository
`git pull <repository link>`	⟶	Incorporates changes from a remote repository into the current branch.
`git clone`	⟶	Downloads existing code from a remote repository
`git stash`	⟶	Used to remove uncommited files temporarily
`git reset`	⟶	Undoes all the commits after the specified commit and preserves the changes locally.
`git rm <file-name>`	⟶	Removes file from the directory

@sajid_curious

Sajid Mohammed

#JavaScript-with-JC

```
1  console.log("start");
2
3  const promise = new Promise((resolve) ⇒ {
4    console.log(1);
5    resolve(2);
6    console.log(3);
7  });
8
9  promise.then((result) ⇒ {
10   console.log(result);
11 });
12
13 console.log("end");
14
15 👍A) start end 1 3 2      💡B) start 1 3 end 2
16 💖C) start end 1 2 3      😀D) start 1 end 2 3
17
```

Jayesh Choudhary
@JavaScript-with-JC

JavaScript

VAR vs LET vs CONST

	var	let	const
Reassignment	✅	✅	❌
Redeclaration	✅	❌	❌
Function Scope	✅	✅	✅
Block Scope	❌	✅	✅
Hoisting	✅	✅	✅
TDZ (Temporal Dead Zone)	❌	✅	✅
Global object property (window)	✅	❌	❌

Pradeep Pandey

CSS Units - Cheatsheet

CSS Units determines the size of a property you're setting for an element of its content

1. Absolute Units

Units that are "absolute" are the same size regardless of the parent element or window size .

Absolute Units		Description
px	→	1px = 1/96th of 1 inch (96px = 1 inch)
pt	→	1pt = 1/72th of 1 inch
mm	→	1mm = 1/10th of 1cm
cm	→	Centimeters
pc	→	12pt=1pc
in	→	Inches

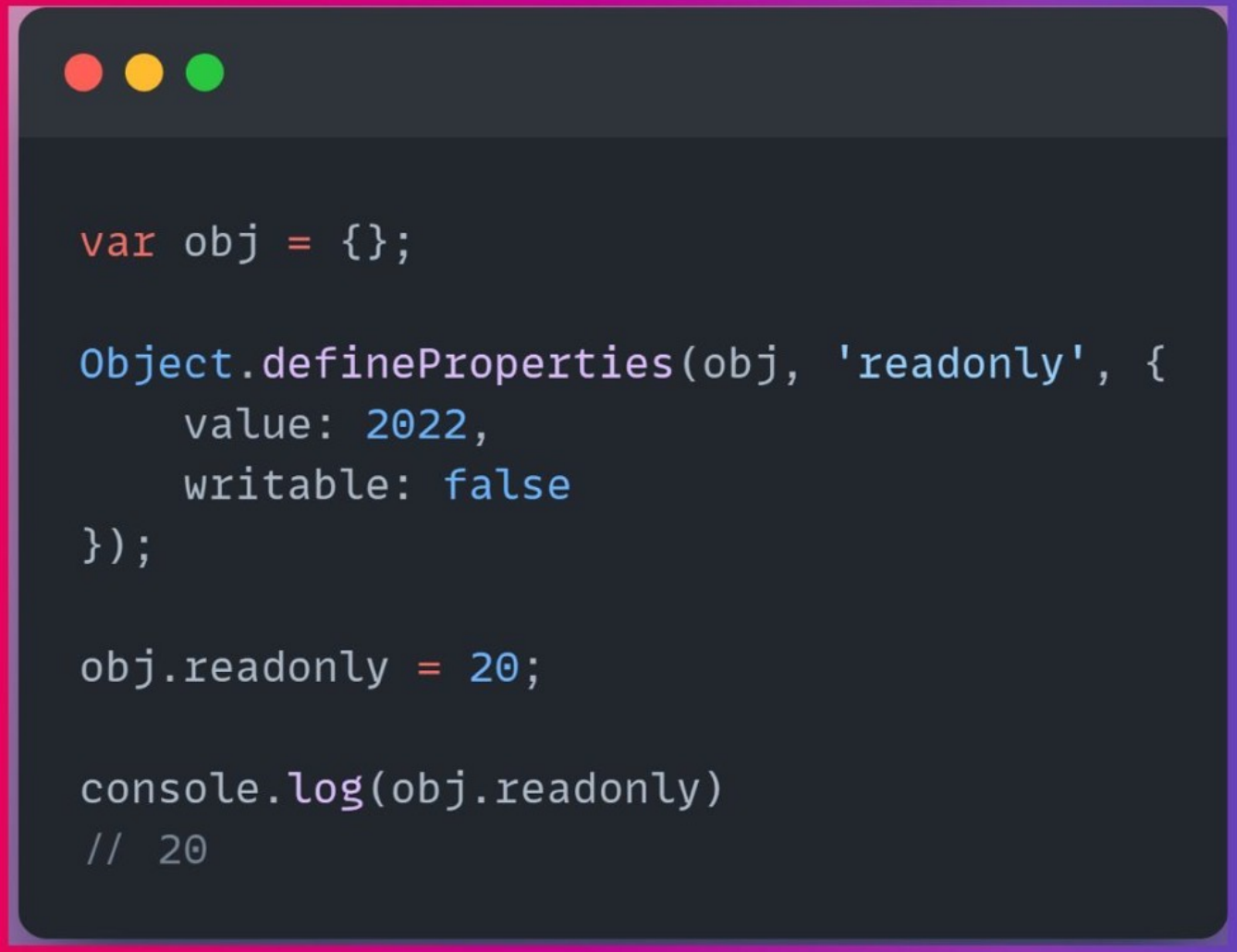
2. Relative Units

Relative units are useful for styling responsive sites because they scale relative to the parent or window size (depending upon the unit)

Relative Units		Description
%	→	Relative to the parent element
em	→	Relative to the font size of the element
rem	→	Relative to the font size of the root element
ch	→	Relative to the width of the "0" (zero)
vh	→	Relative to 1% of the viewport's height.
vw	→	Relative to 1% of the viewport's width.
vmin	→	Relative to 1% of the viewport's smaller dimension.
vmax	→	Relative to 1% of the viewport's larger dimension.

Pradeep Pandey

OBJECT PROPERTIES IN JAVASCRIPT



```
var obj = {};

Object.defineProperties(obj, 'readonly', {
    value: 2022,
    writable: false
});

obj.readonly = 20;

console.log(obj.readonly)
// 20
```

in Nageshwar Nag

CSS Tip

FILL TEXT WITH ANY BACKGROUND

HTML

```html
<h1>Green Grass</h1>
```

CSS

```css
h1 {
  background-image: url('/images/grass.jpg');
  -webkit-background-clip: text;
  -webkit-text-fill-color: transparent;
}
```

✅ Text is selectable and accessible

✅ Works in all modern browsers

CODE WITH SHRIPAL

HTML Basics Cheatsheet Part 1

Basic Structure

<!DOCTYPE> ⟶ Version of HTML
<html> ⟶ HTML document
<head> ⟶ Page Information
<body> ⟶ Page Contents

Page information

<meta /> ⟶ Metadata
<title> ⟶ Title
<link /> ⟶ Link to resource
<script> ⟶ Script resource

Lists

 ⟶ Ordered list
 ⟶ Unordered list
 ⟶ List item

Links

<a href=""> ⟶ Page link
<a href="mailto:"> ⟶ Email link
<a name="name"> ⟶ Anchor link
<a href="#name"> ⟶ Link to anchor

Forms

<form> ⟶ Form
<label> ⟶ Input label
<input /> ⟶ Form input
<select> ⟶ Drop-down box
<optgroup> ⟶ Group of options
<option> ⟶ Drop-down options
<textarea> ⟶ Large text input
<button> ⟶ Button

HTML5 Semantic Tags

<article> ⟶ Article
<details> ⟶ Details of an element
<footer> ⟶ Footer for a section or page
<header> ⟶ Header for a section or page
<mark> ⟶ Marked text
<section> ⟶ Section of page or article
<figure> ⟶ Group of media content and their caption
<aside> ⟶ Outside the main flow of the narrative
<nav> ⟶ Navigation links

@sajid_curious
Sajid Mohammed

3 Ways to Writing Functions In JavaScript

1) Function Declaration

```
function sum(a, b) {
  return a + b;
}
```

2) Function Expression

```
const sum = function(a, b) {
  return a + b;
}
```

3) Arrow Function

```
const sum = (a, b) ⇒ {
  return a + b;
}
```

Sham Gurav

CSS Selectors Cheatsheet Part 1

Simple selectors

Element Selector ⟶ Selects HTML elements based on the element name.

```css
p{
  color: white;
}
```

Id Selector ⟶ Uses the id attribute to select a specific element.

```css
#fname {
  width: 50vw;
}
```

Class Selector ⟶ Uses the class attribute to select a specific element.

```css
.container {
  padding: 3vh 1vw 3vh 1vw;
}
```

Universal Selector ⟶ Selects all HTML elements on the page.

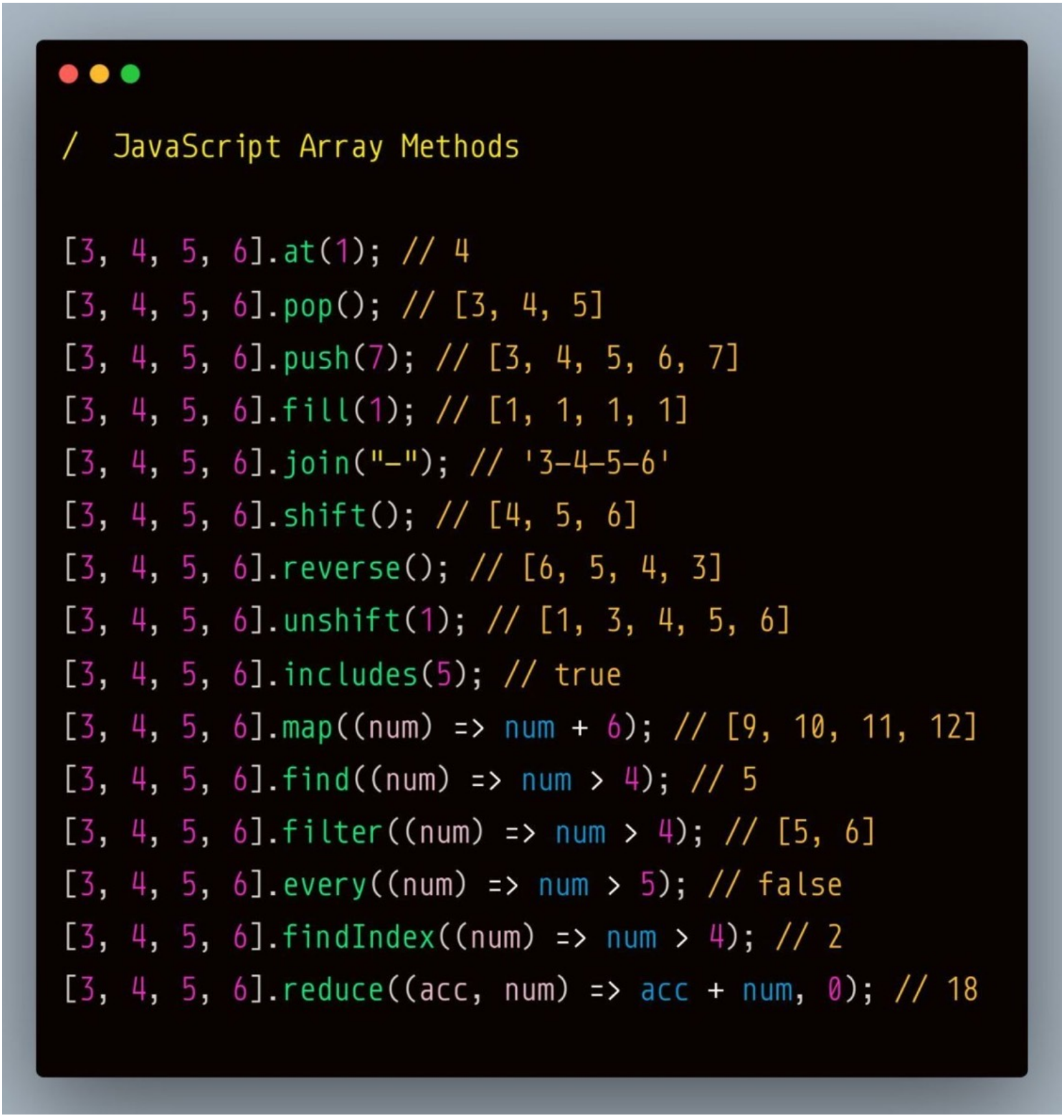
```css
*{
  padding: 50px;
}
```

Grouping Selector ⟶ Selects all the HTML elements with the same style

```css
#item1, #item2, #item3{
  align-self: center;
}
```

@sajid_curious

Sajid Mohammed


/ JavaScript Array Methods
[3, 4, 5, 6].at(1); // 4
[3, 4, 5, 6].pop(); // [3, 4, 5]
[3, 4, 5, 6].push(7); // [3, 4, 5, 6, 7]
[3, 4, 5, 6].fill(1); // [1, 1, 1, 1]
[3, 4, 5, 6].join("-"); // '3-4-5-6'
[3, 4, 5, 6].shift(); // [4, 5, 6]
[3, 4, 5, 6].reverse(); // [6, 5, 4, 3]
[3, 4, 5, 6].unshift(1); // [1, 3, 4, 5, 6]
[3, 4, 5, 6].includes(5); // true
[3, 4, 5, 6].map((num) => num + 6); // [9, 10, 11, 12]
[3, 4, 5, 6].find((num) => num > 4); // 5
[3, 4, 5, 6].filter((num) => num > 4); // [5, 6]
[3, 4, 5, 6].every((num) => num > 5); // false
[3, 4, 5, 6].findIndex((num) => num > 4); // 2
[3, 4, 5, 6].reduce((acc, num) => acc + num, 0); // 18

HTML preload Attribute

The preload attribute specifies if and how the author thinks that the media file should be loaded when the page loads.

The preload attribute can be used on <audio> and <video> elements.

```
<audio controls preload="none">
  <source src="audioFile.ogg" type="audio/ogg">
</audio>
```

Note: The preload attribute is ignored if autoplay is present. The specification does not force the browser to follow the value of this attribute; it is a mere hint.

in Sham Gurav

JavaScript String Method

```
const str = 'Hello World';

str.length; // 11

str.charAt(4); // o

str.endsWith('d'); // true;

str.includes('Worlds'); // true;

str.indexOf(); // 6

str.repeat(2); // "Hello WorldHello World"

str.replace('world', 'Pradeep'); // "Hello Pradeep"

str.slice(6, 10); // "worl"

str.split(' '); // ["Hello , World"]

str.startsWith('Hello'); // true

str.substring(6, 12); // "World";

str.substr(1, 4); // "ello";

str.toLowerCase(); // "hello world"

str.toUpperCase(); // "HELLO WORLD"

'     Hello World!     '.trim(); // "Hello World"
```

Pradeep Pandey

```
let map = new Map();              // Creates a Map

console.log(typeof(map));         // object

map.set("Inception", 2010);       // Sets the value for a key in a Map
map.set("Avatar", 2009);
map.set("The Matrix", 1999);

map.get("Avatar");                // 2009

map.size;                         // 3

map.delete("Avatar");             // The delete() method removes a Map element

map.has("The Matrix");            // true

map.clear();                      // clear() method removes all the elements from a Map

map.forEach((v, k) => {
    console.log(k, " : ", v);     // Invokes a callback for each key/value pair in a Map
} );

let text2 = "";
for (const x of map.entries()) {
  text2 += x;
}           // Returns an iterator object with the [key, value] pairs in a Map

let text3 = "";
for (const x of map.keys()) {
  text3 += x;
}        // Lists all keys, keys() Returns an iterator object with the keys in a Map

let text1 = "";
for (const x of map.values()) {
  text1 += x;
}       //  values() Returns an iterator object of the values in a Map  : 1999 2009
```

Remove Dublicates from Array – JavaScript

```
// Using Set method
let arr = ['a', 'b', 'a', 'c', 'c'];
let uniques = Array.from(new Set(arr));
console.log(uniques); // ["a","b","c"]

// Using array filter() method
let arr2 = ['a', 'b', 'a', 'c', 'c'];
let uniques2 = arr2.filter((value, index, arr) => {
  return index == arr.indexOf(value);
})
console.log(uniques2); // ["a","b","c"]
```

Creator: Ankit Mishra

How to Sort JavaScript Object by Key

```
function sortObj(obj) {
  return Object.keys(obj).sort().reduce((result, key) => {
    result[key] = obj[key];
    return result;
  }, {});
}
let obj = {g : 100, a: 2, c: 300};
let sortedObj = sortObj(obj);
console.log(sortedObj); // {"a": 2, "c": 300, "g": 100}
```

Creator: Ankit Mishra

Center a Div Using Flexbox

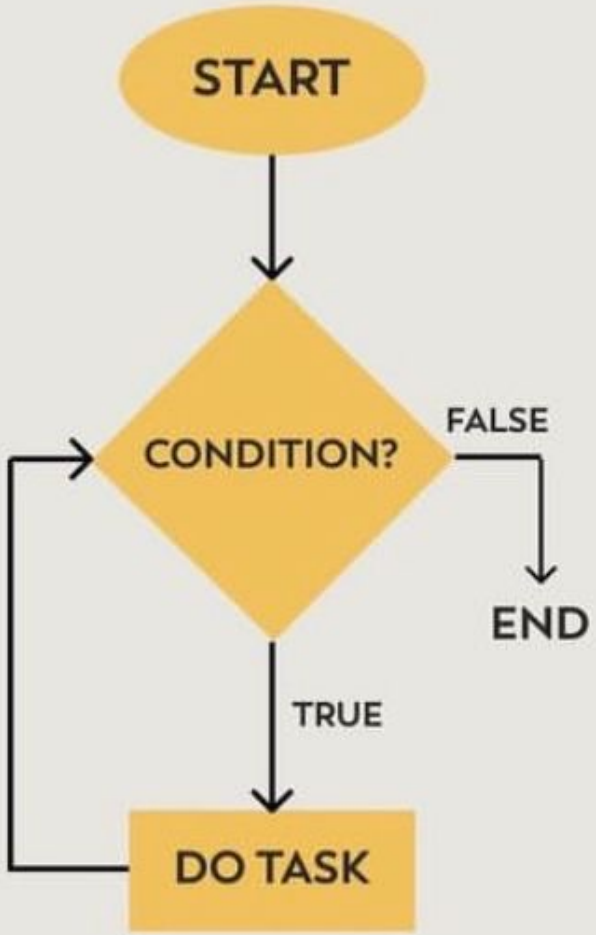
The Flexible Box Layout Module makes designing a flexible responsive layout structure easier without using float or positioning.

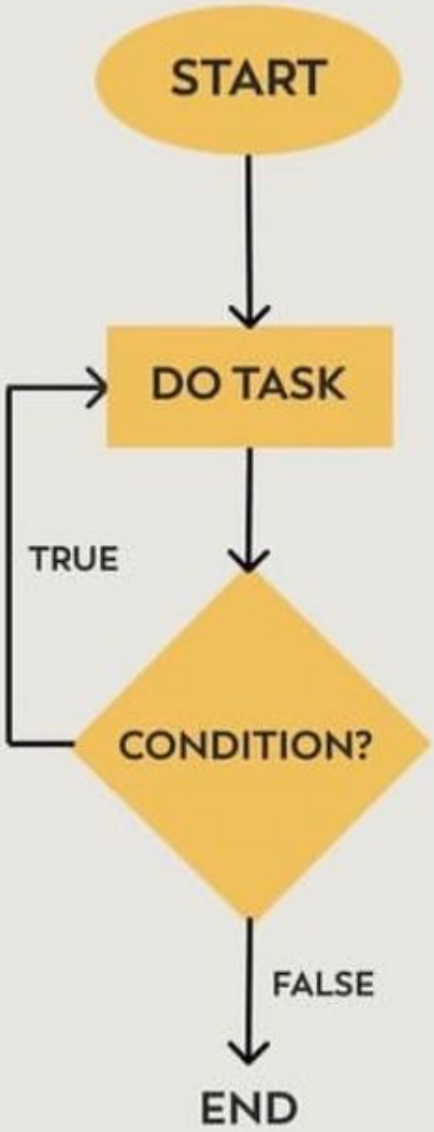
Applying the following properties to .parent will center .child horizontally and vertically.

```
.parent {
  display: flex;
  align-items: center;
  justify-content: center;
}
```

in Sham Gurav

WHILE VS DO-WHILE





Techie Programmer

#JavaScript-with-JC

```
1  function Person(name, age) {
2    this.name = name;
3    this.age = age;
4
5    this.getInfo = function () {
6      return this.name;
7    };
8  }
9
10 Person.prototype.getInfo = function () {
11   return `${this.name} ${this.age}`;
12 };
13
14 const jayesh = new Person("JC", 24);
15 console.log(jayesh.getInfo());
16
17 👍A) JC                    💡B) undefined
18 💖C) JC 24                  😃D) TypeError
19
```

Jayesh Choudhary
@JavaScript-with-JC

Add a CSS class to an element using JavaScript

The add() method of the classList property can be used to add a CSS class from an HTML element.

For example, the following code adds the info class to the class list of the div element with the id content.

```
let div = document.querySelector('#content');

div.classList.add('info');
```

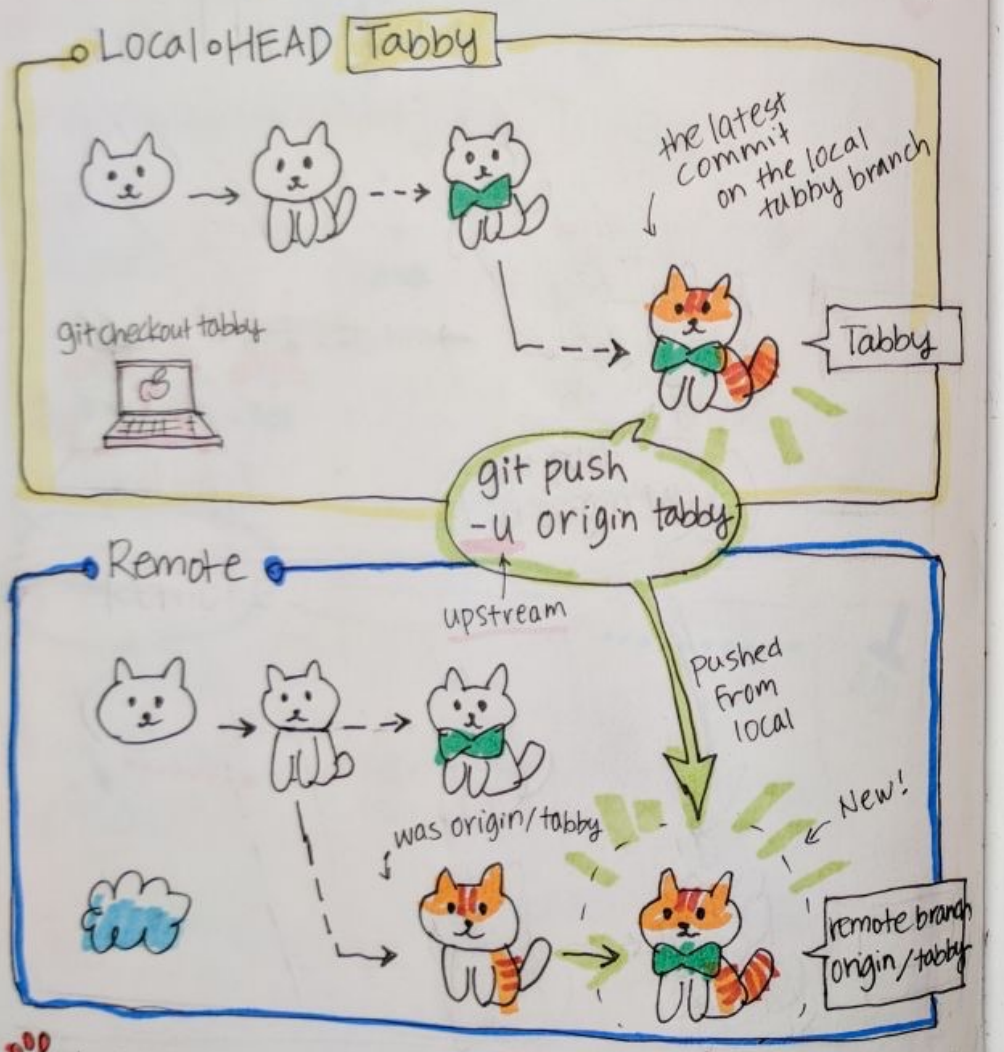
Sham Gurav

git Push

puss

@girlie_mac

♥ git push transfers your commits from your local repo to a remote repo!



Now the tabby branch on local + remote are sync'd!

♥ git push

git push <remote> <branch>

e.g. origin e.g. tabby

push the specified branch w/ all commits

git push -u <remote> <branch>

↳ = --set-upstream

the -u flag sets up the association between your branch + the remote branch explicitly.
you don't need it once you've done!

git push -f <remote> <branch>

↳ --force

force the push, even if it results in a non-fast-foward merge.
just ignore + push!

git push --all <remote>

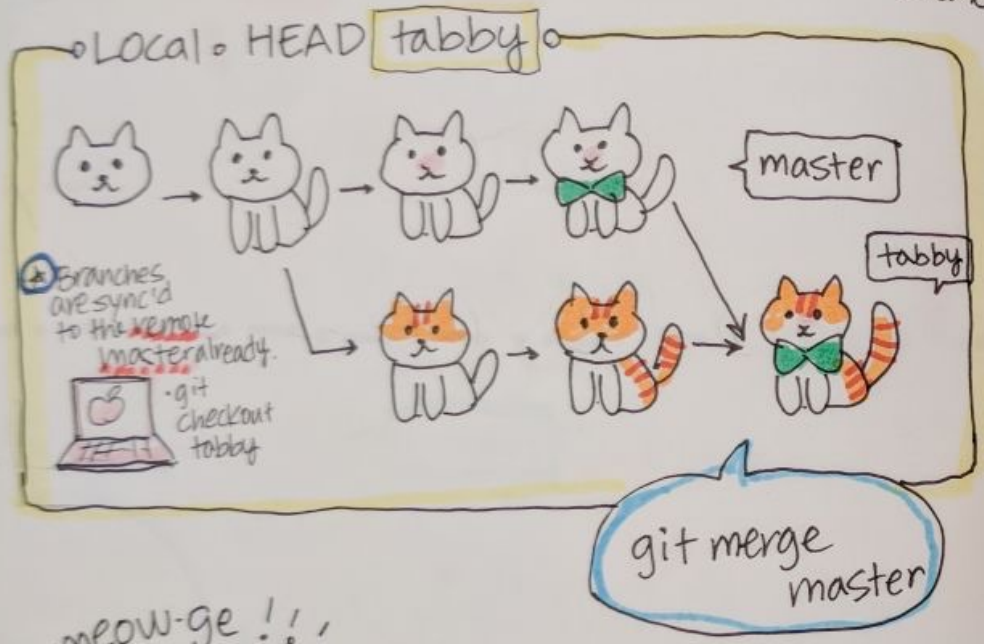
push all of your local branches!

git merge

meow! ge

@girlie_mac

- git merge incorporates changes into the current branch.



git meow-ge !!!

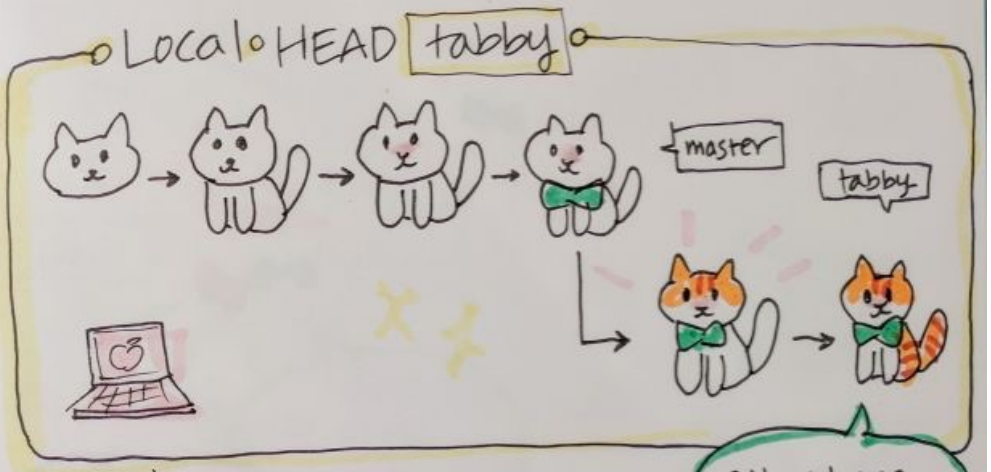
- ★ merge is like having 2 parents & 1 resulting child !
- ★ rebase adds all new △s on top of one parent.

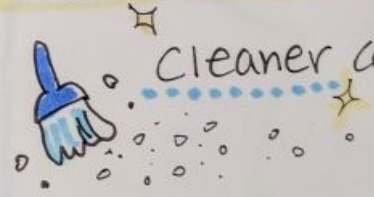
PAWSOME!

git rebase

@girlie_mac

- git rebase moves a branch from one commit to another.





Cleaner commit history!

git rebase master

If you want to rebase from the remote master instead of local. do either:

```
$ git fetch origin
$ git rebase origin/master
```

or

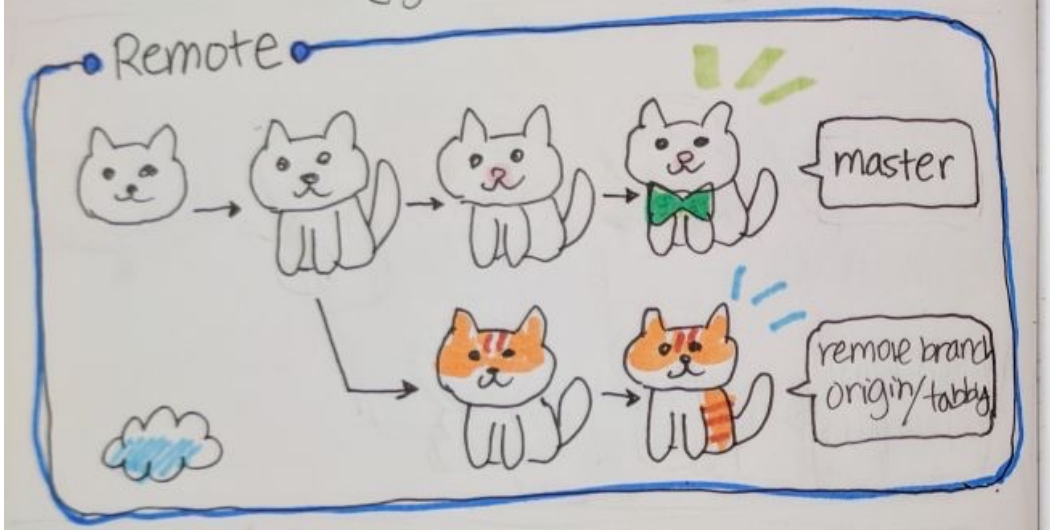
```
$ git pull --rebase origin master
```

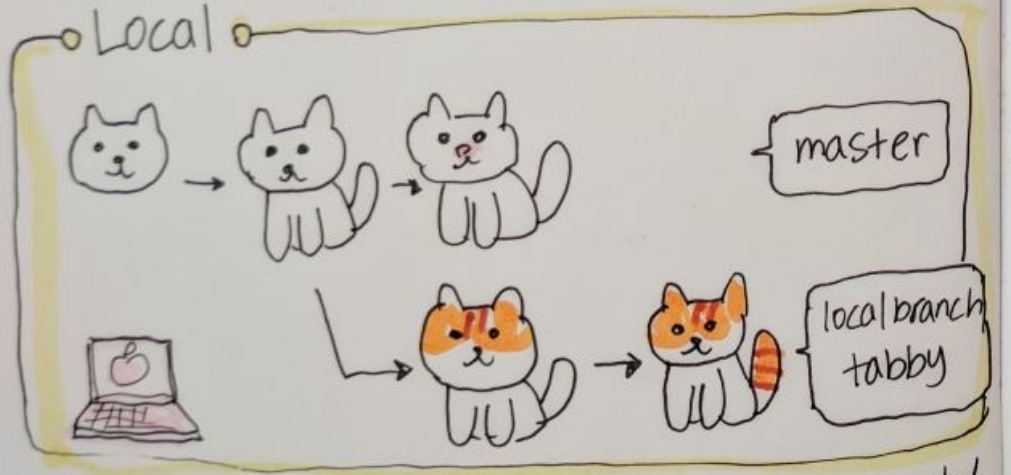
git Pull

purr

@girlie_mac

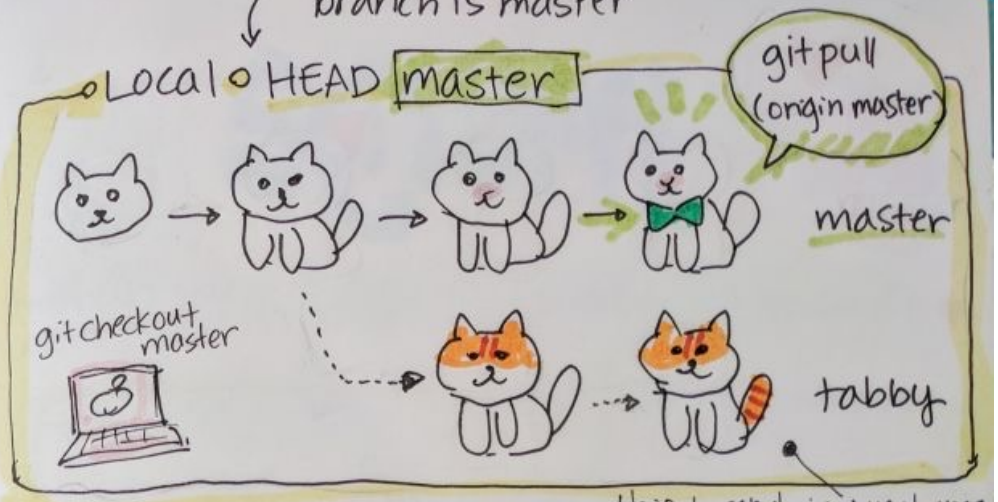
git pull updates your current HEAD branch w/ the latest Δs from remote





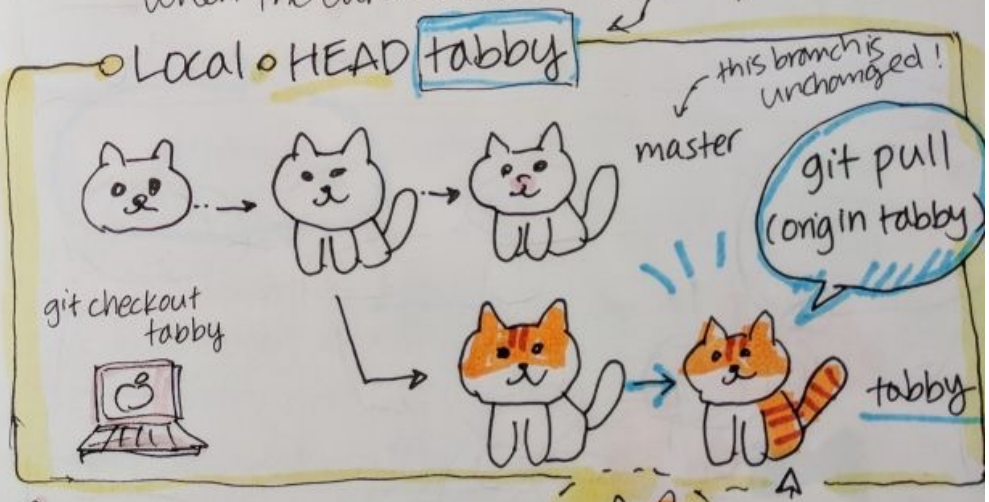
Now I'm going to 'git pull' to update a branch!

When the current branch is master



This is same as:
- git fetch origin
- git merge origin/master

When the current branch is tabby

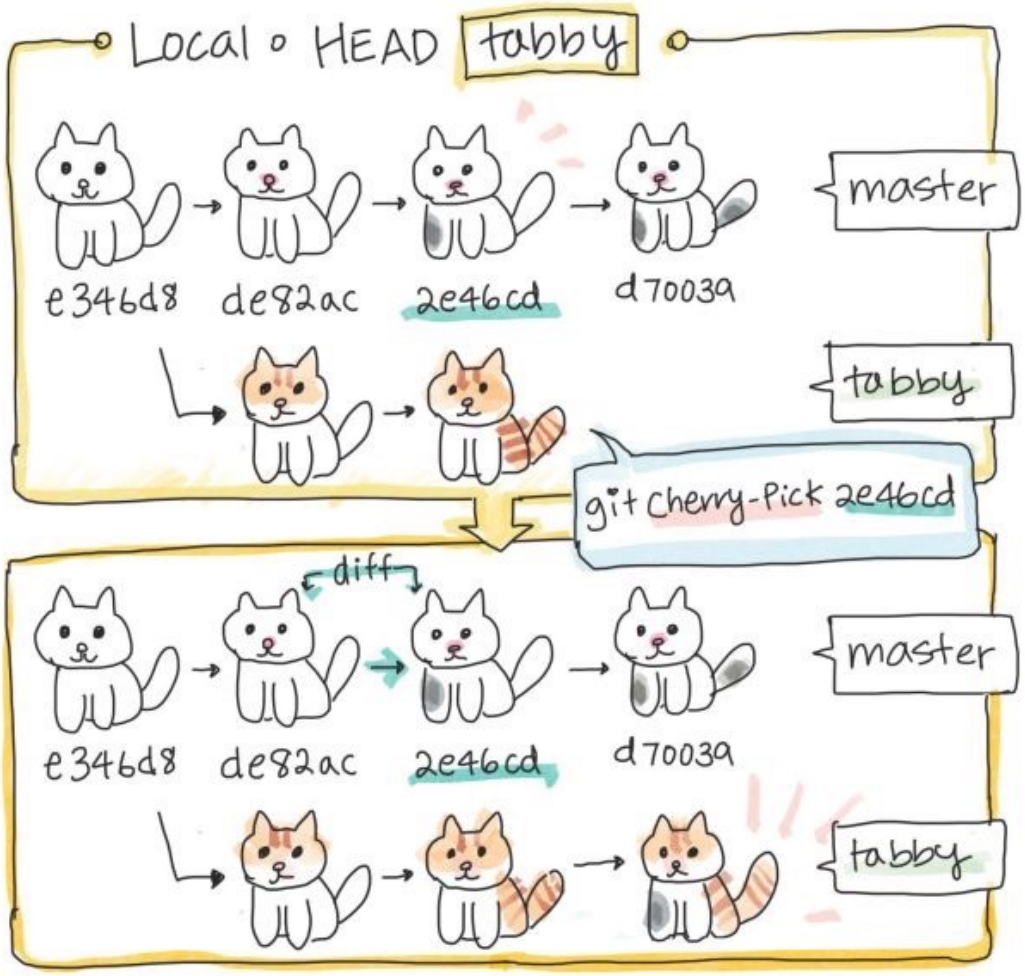


Same as:
- git fetch origin
- git merge origin/tabby

git cherry-pick

@girlie_mac

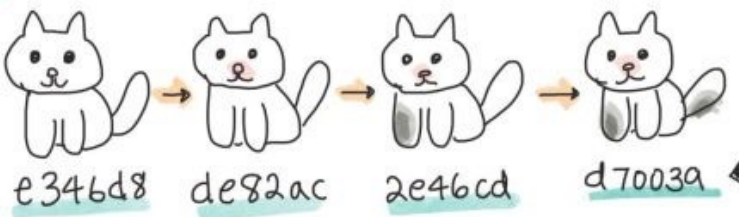
♥ git cherry-pick lets you grab some commits from one branch & apply it into another branch!



git log

♥ git log lets you view the commit history

@girlie_mac



♥ git log prints out

```
Commit e346d8
Author: Little-princess Leia <leia@kitty.cat>
Date : Sun Sept 23 17:30:42 2018 -0700
    Add body
commit de82ac
Author; Jamie <jam@kitty.cat>
Date :
....
```

git cherry-pick this SHA into the tabby branch

Simpler log with:

♥ git log --oneline

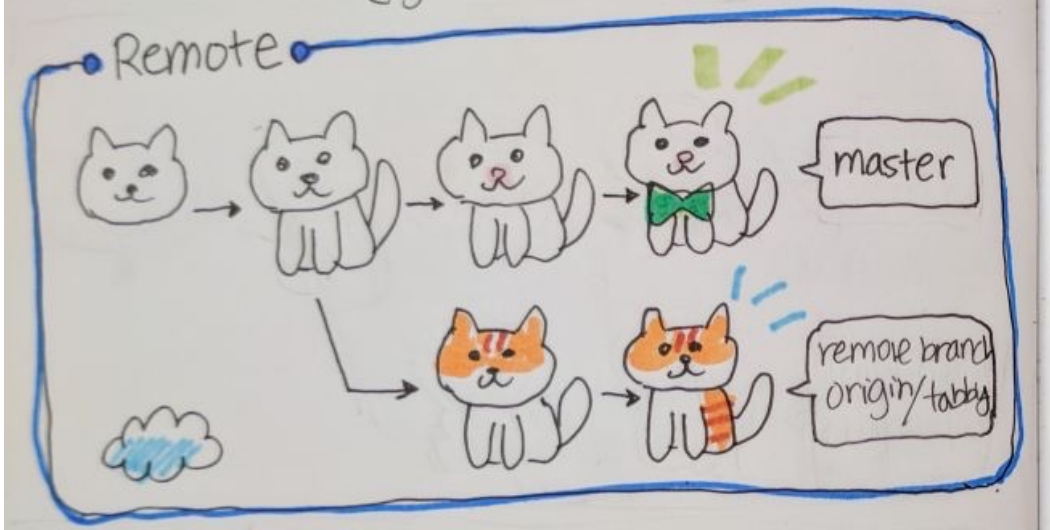
```
e346d8 Add body
de82ac Edit nose
2e46cd Add gray dot on leg
```

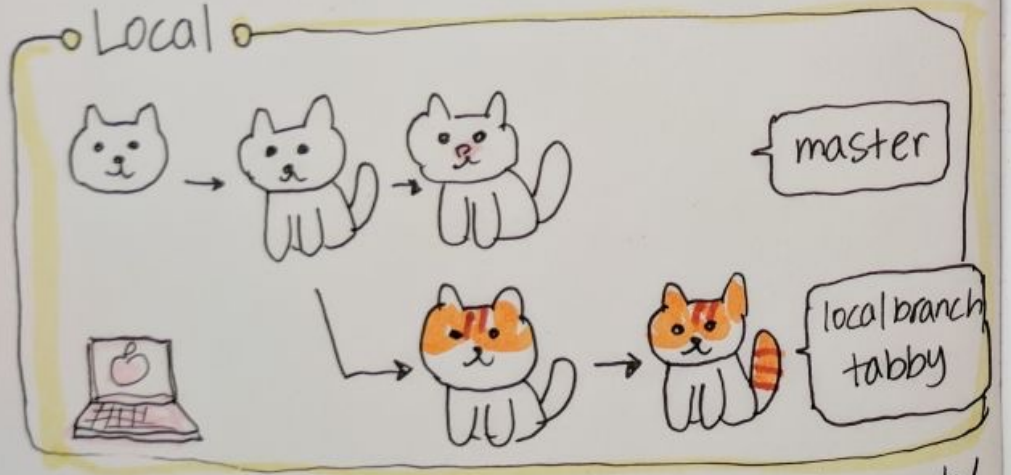
git Pull

purr

@girlie_mac

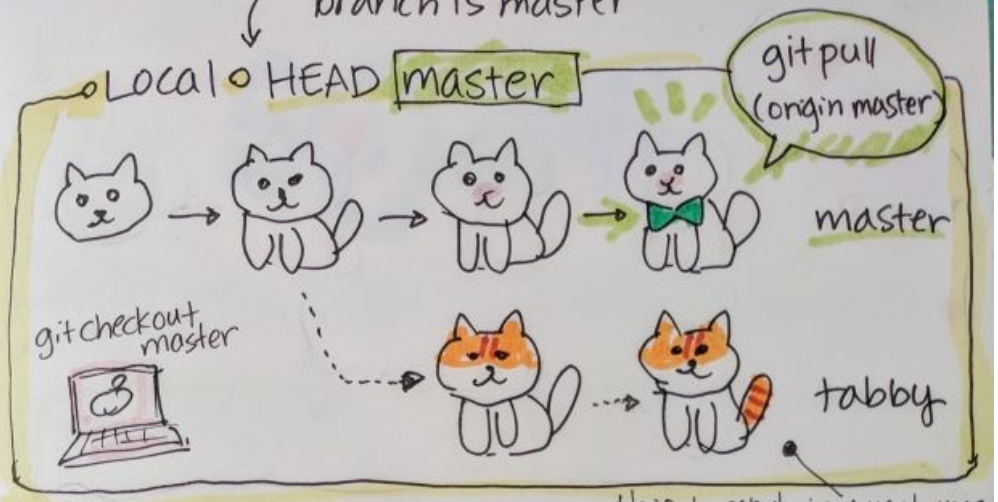
git pull updates your current HEAD branch w/ the latest Δs from remote





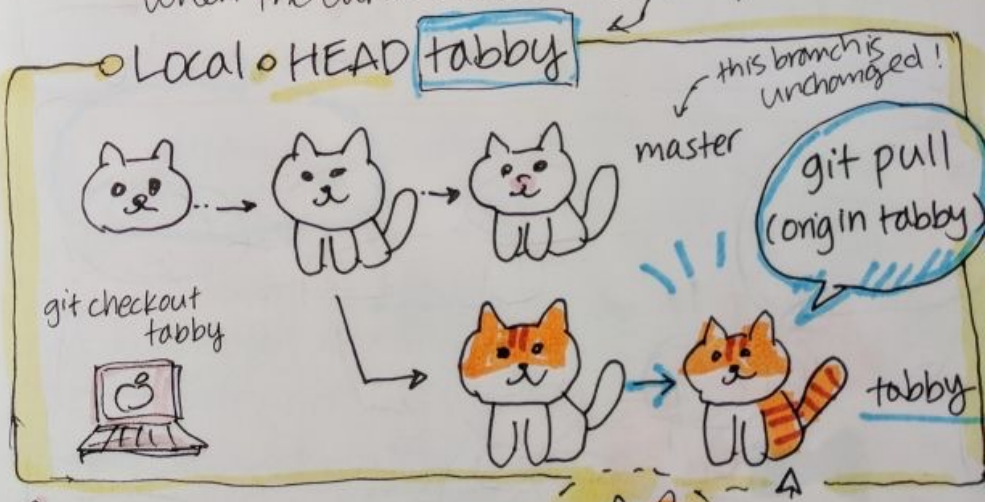
Now I'm going to 'git pull' to update a branch!

When the current branch is master



This is same as:
- git fetch origin
- git merge origin/master

When the current branch is tabby



Same as:
- git fetch origin
- git merge origin/tabby

```
const name = 'Ayush';

name.length                          // 5

name.slice(2,4)                      // ush

name.slice(2,4)                      // us

name.slice(-2)                       // sh

name.substr(2, 1)                    // u

name.replace('yush', 'jay')          // Ajay

name.toUpperCase()                   // AYUSH

name.toLowerCase()                   // ayush

name.concat(' ', 'Thakur')           // Ayush Thakur

'        Ayush'.trim()               // Ayush

name.padStart(7, 'A')                // AAAyush

name.padEnd(7, 'A')                  //  AyushAA

name.charAt(3)                       //  s

name.charCodeAt(3)                   // 115

name[3]                              //  s

name.split('')                       // [ 'A', 'y', 'u', 's', 'h' ]

name.indexOf('u')                    //  u

'AyushA'.lastIndexOf('A')            //  5

name.search('s')                     //  3

name.includes('s')                   //  true

name.startsWith('s')                 //  false

name.endsWith('h')                   //  true
```