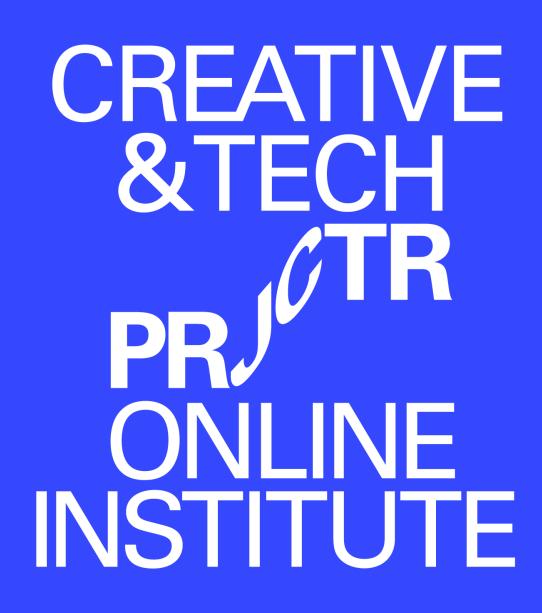


Продвинута робота з масивами та об'єкт Date

2023

адженда



01 / Продвинута робота з масивами

02 / Об'єкт Date

03 / Деструктурізація

2023

prjctr.com



Продвинута робота з масивами

2023

prjctr.com

Масиви об'єктів

Масив об'єктів — це структура даних, де кожен елемент масиву є об'єктом.

```
const students = [
      { id: 1, name: "Іван", age: 20 },
      { id: 2, name: "Олена", age: 22 },
      { id: 3, name: "Микола", age: 21 }
];
```



Meтод for Each

Meтод forEach використовується для ітерації по елементах масиву.

Він приймає функцію зворотнього виклику (callback) як аргумент і виконує цю функцію для кожного елемента масиву.



Важливо відзначити, що for Each просто виконує функцію для кожного елемента і не повертає новий масив.

```
const numbers = [1, 2, 3, 4, 5];
numbers.forEach((element, index, array) \Rightarrow {
    console.log(`Element at index ${index} is ${element}`);
});
```

Иетод тар

Метод тар використовується для створення нового масиву, шляхом виконання функції для кожного елемента поточного масиву.

Відмінність map від методів як forEach полягає в тому, що map повертає новий масив на основі результатів виконання функції зворотнього виклику.

```
CREATIVE & TECH / TR
PROBLINE
ONLINE
INSTITUTE
```

```
const numbers = [1, 2, 3, 4];
const doubled = numbers.map(num \Rightarrow num * 2);
console.log(doubled); // [2, 4, 6, 8]
```

Meтод find

Метод find використовується для знаходження першого елемента масиву, який відповідає визначеній умові. Ця умова визначається функцією зворотнього виклику, яку ви передаєте як аргумент методу find.



find повертає перший елемент, який відповідає умові, або undefined, якщо жоден елемент не відповідає умові.

```
const numbers = [4, 7, 12, 5, 9];
const found = numbers.find(num \Rightarrow num > 10);
console.log(found); // 12
```

Mетод findIndex

Метод findIndex використовується використовується для знаходження індексу першого елемента в масиві, який відповідає визначеній умові. Ця умова визначається функцією зворотнього виклику, яку ви передаєте як аргумент методу findIndex.



findIndex повертає індекс першого елемента, який відповідає умові, або -1, якщо жоден елемент не відповідає умові.

```
const numbers = [4, 7, 12, 5, 9];
const index = numbers.findIndex(num \Rightarrow num > 10);
console.log(index); // 2
```

Mетод filter

Метод filter є методом масиву, який створює новий масив із усіх елементів, які проходять умову, задану у функції зворотнього виклику.



filter повертає новий масив, що складається з елементів, які відповідають визначеній умові.

```
const numbers = [1, 2, 3, 4, 5, 6];
const evens = numbers.filter(num \Rightarrow num % 2 == 0);
console.log(evens); // [2, 4, 6]
```

Mетоди some / every

Метод some перевіряє, чи хоча б один елемент масиву відповідає заданій умові, яка визначається у функції зворотнього виклику.

Метод every перевіряє, чи всі елементи масиву відповідають певній умові, яка визначається у функції зворотнього виклику.

```
const numbers = [2, 4, 6, 8, 10];
const hasEven = numbers.some(num \Rightarrow num % 2 \Rightarrow 0);
console.log(hasEven);

const areAllEven = numbers.every(num \Rightarrow num % 2 \Rightarrow 0);
console.log(areAllEven);
```



Metoд sort

Метод sort використовується для сортування елементів масиву на місці та повертає відсортований масив.

Meтод sort сортує масив на місці, тобто змінює оригінальний масив.

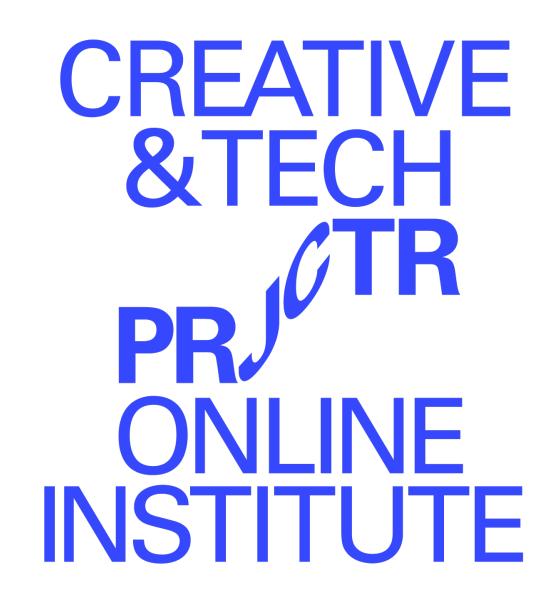
За замовчуванням елементи сортуються як рядки. Для кастомного порядку сортування можна передати функцію порівняння як аргумент.

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



Metoд sort

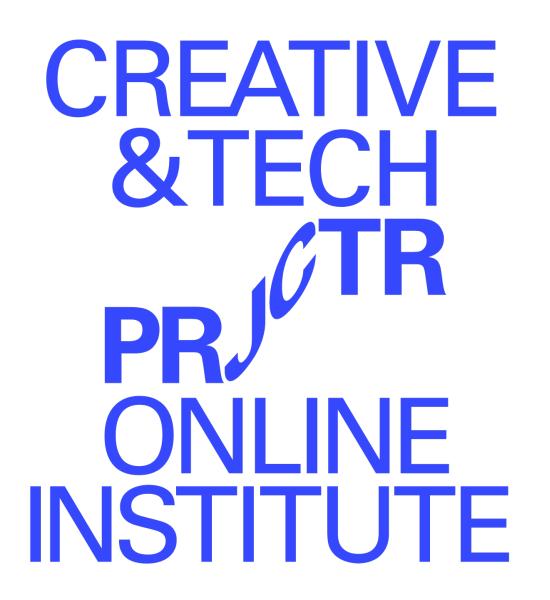
Насправді від функції порівняння потрібно будь-яке позитивне число, щоб сказати «більше», і негативне число, щоб сказати «менше».



```
const numbers = [10, 5, 40, 25, 100];
numbers.sort((a, b) ⇒ a - b);
console.log(numbers); // [5, 10, 25, 40, 100]
```

Метод reduce

Метод reduce використовується для обробки масиву елемент за елементом, накопичуючи результат у "акумуляторі", і в результаті повертає єдине значення.



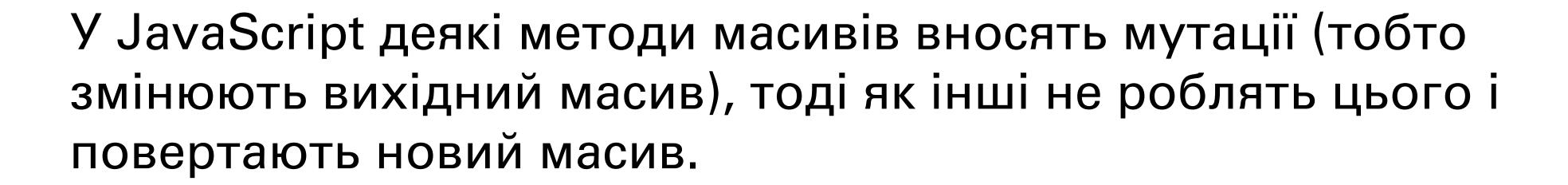
Він використовується для обчислення якогось єдиного значення на основі всього масиву.

При виклику функції, результат її виклику на попередньому елементі масиву передається як аргумент.

```
const numbers = [1, 2, 3, 4, 5];
const sum = numbers.reduce((accumulator, currentValue) ⇒ accumulator + currentValue, 0);
console.log(sum); // 15
```

Мутації

Мутація - визначає зміни, які вносяться безпосередньо у вихідний об'єкт чи масив, а не створюють нову копію цього об'єкта або масиву.



Методи масивів, які вносять мутації: push, splice, reverse, sort.

Методи масивів, які не вносять мутації (не змінюють вихідний масив): slice, concat, map, filter, reduce, find.

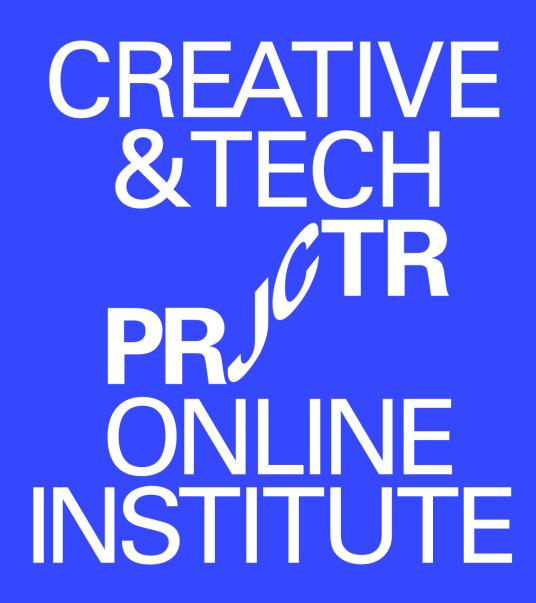


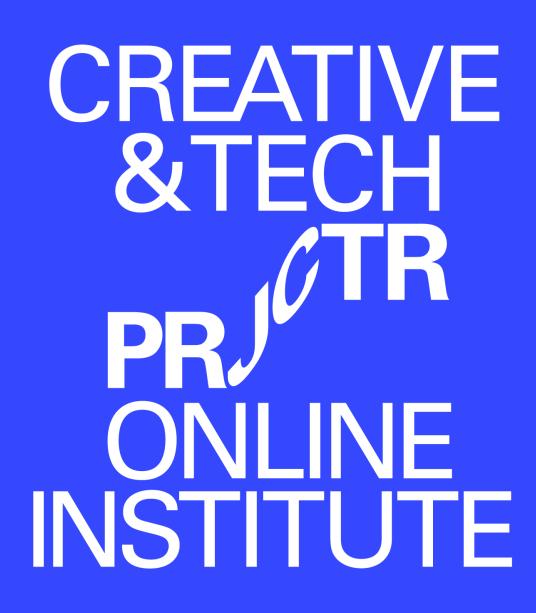
Мутації

Якщо вам важливо уникнути мутації вихідного масиву (наприклад, для того щоб написати "чисту функцію"), то важливо знати, які методи змінюють вихідний масив, і вибирати немутуючі методи або створювати копії масивів перед їх зміною.



```
const original = [1, 2, 3, 4];
const copy = original.slice();
```





Oб'ekt Date

2023

prjctr.com

Об'єкт Date

Об'єкт Date y JavaScript використовується для роботи з датами та часом.

За замовчуванням, дата і час вимірюються в мілісекундах, що починаються з 1 січня 1970 року UTC, яку зазвичай називають "Епохою Unix".

```
const now = new Date();
console.log(now);
```



Параметри

Об'єкт Date може приймати декілька варіантів параметрів для того щоб вказати спеціфічну дату.

- Рядковий параметр
- Числовий параметр
- Кількість мілісекунд

```
CREATIVE & TECH / TR / TR / PR / ONLINE INSTITUTE
```

```
const dateFromString = new Date("December 17, 1995 03:24:00");
const specificDate = new Date(2020, 4, 15, 14, 56, 0);
const dateFromMilliseconds = new Date(1000000000000);
```

Отримання дати

Об'єкт Date має декілька методів, які дозволяють отримувати конкретні частини дати та часу.

Розглянемо декілька з них:



```
const today = new Date();
console.log(today.getFullYear()); // Отримати поточний рік
console.log(today.getMonth()); // Отримати поточний місяць (0-11)
console.log(today.getDate()); // Отримати поточний день місяця (1-31)
```

Встановлення дати

Об'єкт Date має декілька методів, які дозволяють встановлювати конкретні частини дати та часу.

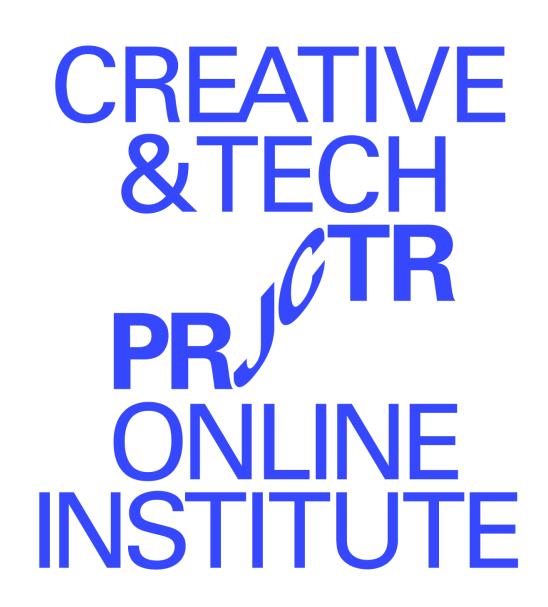
Розглянемо декілька з них:





Конвертація дати в рядок

Ми можемо конвертувати дату у рядок наступним чином:



```
const today = new Date();
console.log(today.toString());  // Повний рядок дати і часу
console.log(today.toDateString());  // Рядок лише з датою
console.log(today.toISOString());  // Дата у форматі ISO
```

Timestamp

"timestamp" зазвичай вказує на кількість мілісекунд, що минула з початкової дати, яка відома як "Епоха Unix" або "Epoch time". Ця початкова дата визначена як 00:00:00 UTC на 1 січня 1970 року.



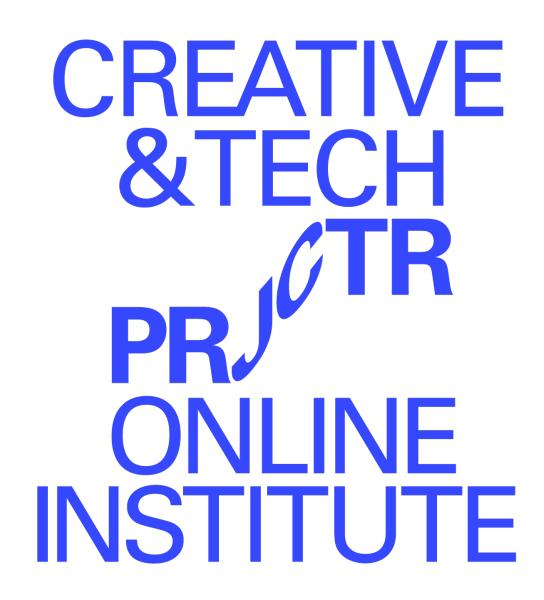
Метод Date.now() повертає поточний timestamp, який представляє собою кількість мілісекунд, що минула від початку Епохи Unix до поточного моменту часу.

```
const timestamp = Date.now(); // Повертає поточний timestamp
const dateFromTimestamp = new Date(timestamp);
console.log(dateFromTimestamp);
```

Timestamp

Ще один метод, який пов'язаний з timestamp — це getTime(). Він повертає timestamp для конкретного об'єкта Date.

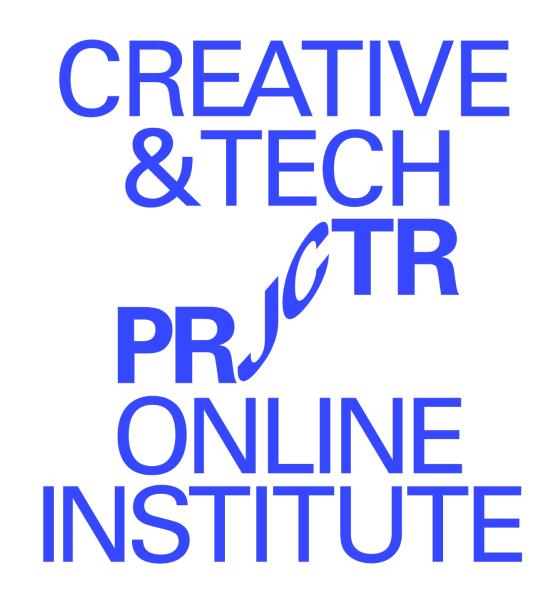
Timestamps у JavaScript корисні, особливо коли вам потрібно порівняти дві дати, зберегти дату у компактному форматі, або конвертувати між часовими зонами та форматами.



```
const someDate = new Date("2023-01-01");
const timestampForSomeDate = someDate.getTime();
console.log(timestampForSomeDate); // Поверне timestamp для 1 січня 2023 року
```

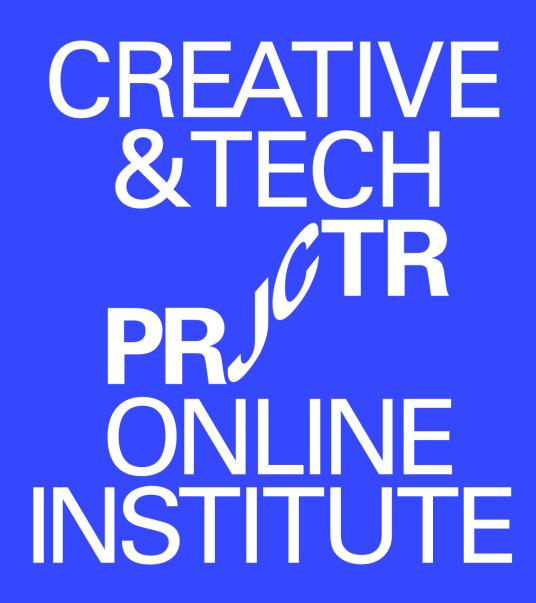
Date.parse

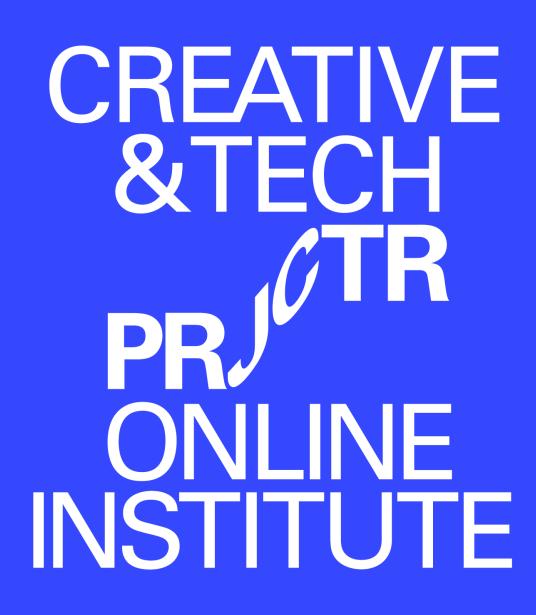
Метод Date.parse розбирає рядкову виставу дати і повертає кількість мілісекунд, що пройшли з 1 січня 1970 00:00:00 по UTC.



Цей метод приймає один аргумент: рядок, що представляє дату. Якщо рядок може бути розпізнаний як валідна дата, метод повертає відповідний timestamp. Якщо рядок не може бути інтерпретований як дата, метод поверне NaN.

```
console.log(Date.parse("2023-01-01Т00:00:00Z"));// Повертає timestamp console.log(Date.parse("March 12, 2022"));// Повертає timestamp для 12 березня 2022 року console.log(Date.parse("Invalid Date String")); // Повертає NaN
```





Деструктуризація

2023

prjctr.com

Деструктуризація, або деструктуруюче присвоєння



Деструктуруюче присвоєння – це спеціальний синтаксис, який дозволяє нам «розпакувати» масиви чи об'єкти у кілька змінних, оскільки іноді вони зручніші.

Деструктуризація також чудово працює зі складними функціями, які мають багато параметрів, значень за умовчанням тощо.

Деструктуризація масивів

Деструктуризація масивів - це процес видобування значень з масивів і присвоєння їх новим змінним. Вона зазвичай використовується, коли ви хочете отримати окремі елементи масиву в якості окремих змінних.



```
let colors = ['red', 'green', 'blue'];
let [firstColor, secondColor] = colors;
console.log(firstColor); // red
console.log(secondColor); // green
```

Деструктуризація об'єктів

Деструктуризація об'єктів дозволяє видобувати значення властивостей об'єктів і присвоювати їх новим змінним.



```
let person = { name: 'John', age: 30 };
let { name, age } = person;
console.log(name); // John
console.log(age); // 30
```

Деструктуризація у функціях

Ви можете деструктуризувати об'єкти або масиви прямо в параметрах функцій.

```
function introduce({name, age}) {
  console.log(`My name is ${name} and I am ${age} years old.`);
const person = {
  name: 'John',
  age: 30
introduce(person);
```

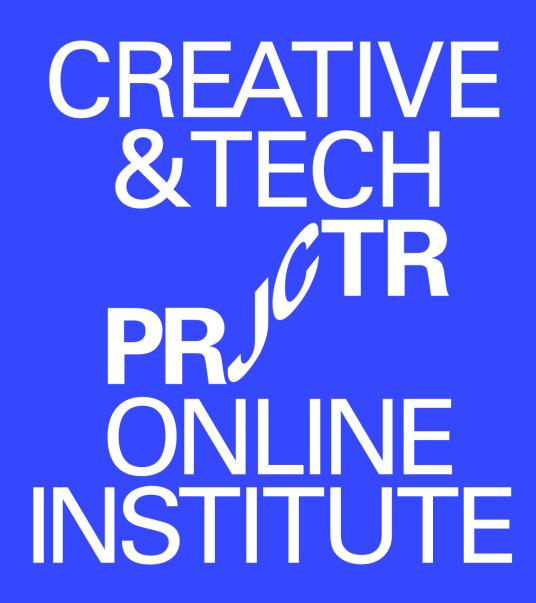


Деструктуризація у методах масивів

Деструктуризація може бути корисною при ітерації по масиву об'єктів або в інших методах:

```
const users = [
  { id: 1, name: 'John' },
  { id: 2, name: 'Jane' },
  { id: 3, name: 'Doe' }
users.forEach(({ id, name }) \Rightarrow {
  console.log(`ID: ${id}, Name: ${name}`);
```





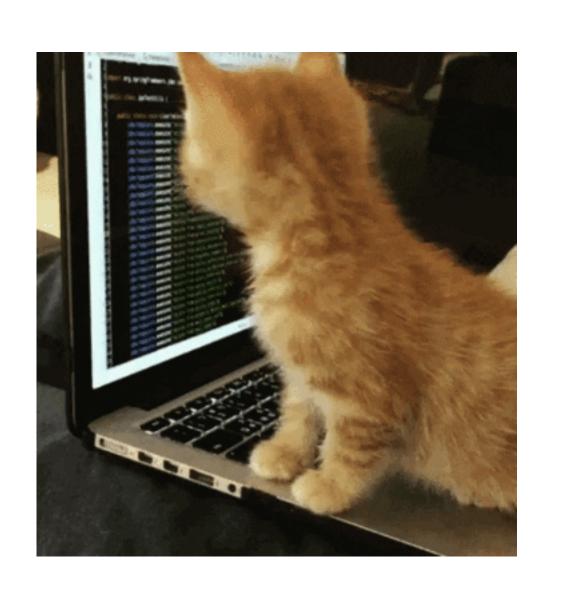
Summary



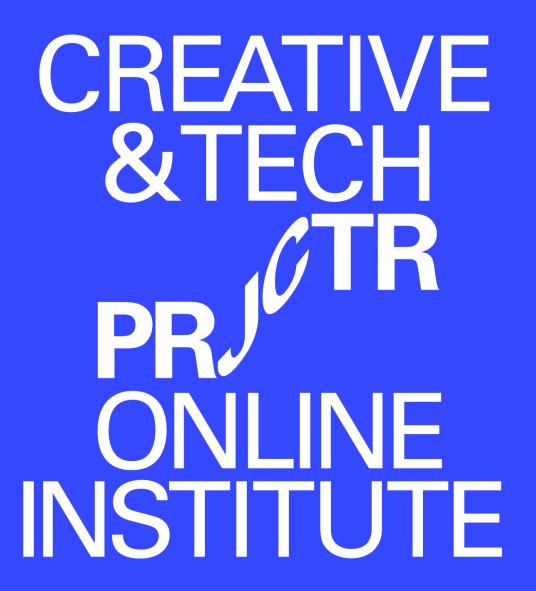
01 / Дізнались про нові методи масивів

02 / Дізнались про об'єкт Date

03 / Познайомились з терміном деструктуризація та розглянули приклади



What's next?



01 / Git Ta GitHub

