Newbie

* usestrict - унеможливлює створення таких змінних num = 5 тобто без ключового слова **let, const** або простого оголошення зміної без значення **let num;**
* What does HTML mean?
* Мова розмітки гіпертексту (HTML) - це стандартна мова розмітки для документів, призначених для відображення у веб-браузері.
* What does CSS mean?
* CSS - це абревіатура від "Каскадні таблиці стилів". CSS - це комп’ютерна мова для розміщення та структурування веб-сторінок (HTML або XML).
* Are HTML and CSS programming languages?
* Так

# HTML

1. What is DOCTYPE and why is it needed?

* Декларація про тип документа HTML, є першим рядком коду, необхідним для кожного документа HTML або XHTML. Декларація DOCTYPE - це інструкція веб-браузеру про те, у якій версії HTML написана сторінка. Це гарантує, що веб-сторінка аналізується однаково для різних веб-браузерів.

1. What is an attribute?

* Атрибут - це специфікація, яка визначає властивість об'єкта, елемента чи файла.

1. What is alt attribute for img tag? Should we use it?

* Необхідний атрибут alt визначає альтернативний текст для зображення, якщо зображення неможливо відобразити.

1. What is title attribute? Should we use it?

* Атрибут Title - це атрибут HTML, який можна додати до будь-якого елемента, але найчастіше він використовується із посиланнями та зображеннями. Це дозволяє надати додаткову інформацію про посилання або зображення. ... Зазвичай веб-браузери відображають атрибут title під час наведення курсора миші на вкладку

1. What are data- attributes used for?([article](https://www.w3schools.com/tags/att_data-.asp))

* Атрибут data- \* використовується для зберігання власних даних, приватних для сторінки чи програми. Атрибут data- \* дає нам можливість вбудовувати власні атрибути даних у всі елементи HTML.

1. What’s the meaning of word semantic in Web?

* Набір ключових фраз для машинного читання

1. What’s the difference between <b> and <strong> tags?

* Обидва теги роблять знячення жирним але <b> це ще семантична мітка яка свідчить про важливість тексту для роботів.

1. What’s the difference between div and section?

* Тег <section> визначає розділи в документі, такі як глави, заголовки, колонтитули або будь-які інші розділи документа. тоді як: Тег <div> визначає розділ або розділ у документі HTML. Тег <div> використовується для групування елементів-блоків для форматування їх у CSS.

1. What is srcset of images?

Змінити зображення коли спрацює media активатор.

* Атрибут srcset визначає URL-адресу зображення для використання в різних ситуаціях. Цей атрибут необхідний, коли <source> використовується в <picture>.
* <picture>, із двома вихідними файлами та резервним зображенням:

<picture>

<source media="(min-width:650px)" srcset="img\_pink\_flowers.jpg">

<source media="(min-width:465px)" srcset="img\_white\_flower.jpg">

<img src="img\_orange\_flowers.jpg" alt="Flowers" style="width:auto;">

</picture>

1. What is viewport?

* Viewport - це видима користувачу область веб-сторінки, то, що може побачити користувач, не вдаючись до прокручування.

1. How to add an image to the web page? ( [img](https://www.w3schools.com/tags/tag_img.asp), [picture](https://www.w3schools.com/tags/tag_picture.asp), [background-image](https://www.w3schools.com/cssref/pr_background-image.asp) [svg](https://www.w3schools.com/html/html5_svg.asp), [canvas](https://www.w3schools.com/graphics/canvas_images.asp) )

# CSS

1. How to add CSS for a webpage?

3 способи вставити CSS на ваші веб-сторінки:

1. За допомогою зовнішнього файлу, на який ви посилаєтесь на своїй веб-сторінці: ...
2. Створюючи блок CSS на самій веб-сторінці; зазвичай вставляється вгорі веб-сторінки між тегами <head> та </head>
3. Вставивши код CSS прямо на сам тег
4. What’s the difference between resetting and normalizing CSS? Which would you choose, and why?

Основна різниця полягає в тому, що:

**resetting CSS** мають на меті видалити всі вбудовані стилі браузера. Стандартні елементи, такі як H1-6, p, strong, em та ін., В кінцевому підсумку виглядають абсолютно схожими, взагалі не маючи прикраси. Потім ви повинні додати всі прикраси самостійно.

**normalizing CSS** має на меті зробити вбудований стиль браузера однаковим у всіх браузерах. Такі елементи, як H1-6, будуть однаково виглядати жирними, більшими у всіх браузерах. Потім вам слід додати лише різницю в оздобленні, яку ви потребуєте в дизайні.

**Я використовую resetting тому що більшість моїх макетів мають унікальний дизайн і стандартні стилі не збігаються з стилями в дизайні.**

Вбудовані стилі мають такі недоліки:

* Труднощі в пошуку ми не можемо їх шукати щоб змінити
* Вбудовані стилі мають найвищу специфіку серед селекторів CSS, тому при змін наприклад шрифту в css він не зміниться в браузері. Доки не використаєм **!important**; Або на початку файлу скинем стилі.

1. What’s the difference between margins and paddings?

* Різниця полягає в тому що margins робить зовнішні відступи від блоку до блоку, а paddings відступи в середині блоку.

1. How can one remove underlined style for a link?

* потрібно задати такі стилі до елементу {**text-decoration: none**}

1. What are media queries? Explain with syntax? ([article](https://www.w3schools.com/Css/css3_mediaqueries.asp))

* Медіа-запити потрібні для відображення різних стилів в залежності від пристрою.

1. What’s selectors specificity? ([article](https://developer.mozilla.org/en-US/docs/Web/CSS/Specificity))

* **Специфічність** - це спосіб, за допомогою якого браузери вирішують, які значення властивостей CSS є найбільш релевантними для елемента і, отже, застосовуватимуться. Чим точніше вказуємо шлях до елемента тим більший пріоритет матимуть такі стилі

<div id="test"> <span>Text</span> </div>

div#test span { color: green; } div span { color: blue; } span { color: red; }

Незалежно від порядку, текст буде зеленим, оскільки це правило є найбільш конкретним

1. What’s the difference between pseudo-element & pseudo-classes in CSS? ([pseudo-class](https://www.w3schools.com/css/css_pseudo_classes.asp))([pseudo-elements](https://www.w3schools.com/css/css_pseudo_elements.asp))

* Псевдоелемент - це елемент, насправді його немає в html документі ми його створюємо в css.
* Псевдокласи схожі на "підроблені" класи, які застосовуються до елементів за певних умов, подібно до того, як ви маніпулюєте класами елементів за допомогою JavaScript.

1. What is tabindex? How and when to use it? ([article](https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/tabindex))

* Атрибут HTML tabindex використовується для управління фокусом клавіатури. Користувач клавіатури, як правило, пересувається через веб-вміст за допомогою клавіші табуляції, переходячи від одного фокусуючого елемента до наступного в послідовному порядку.

1. What do :focus and :active mean?

:focus - це коли елемент може приймати введення - курсор у полі введення або посилання, на яке було встановлено tab.

:active - це коли елемент активується користувачем - час між тим, як користувач натискає кнопку миші, а потім відпускає її.

1. What is outline? Is it better to keep or to remove outline? ([article](https://www.w3schools.com/css/css_outline.asp))

* outline Контур - це лінія, яка намальована навколо елементів поза межами кордонів, щоб елемент «виділився». {outline: #4CAF50 solid 10px;}

1. Explain the position property in CSS & its values. ([article](https://www.w3schools.com/css/css_positioning.asp))
2. Основні властивості **position** та для чого вони існують ?
   * **absolute** - елемент може знаходитися у будь якій точці нашого екрану, і для того щоб його позиціонувати ми використовуємо такі властивості як top, left, right, bottom, крім того елемент який має властивість position absolute він випадає з потоку, тобто він не знаходиться в потоці з іншими елементами
   * **relative** - відносне позиціонування, перевага його над absolute в тому що коли ми його використовуємо елемент залишається в потоці але ми можемо його переміщувати **fixed** - закріпити елемент в певній точці нашого екрану так щоб при скролі сторінки він залишався на тому самому місці
   * **stattik** - це значення по замовчуванню, статичне позиціонування поки елемент не буде мати іншого значення в ньому не будуть працювати допоміжні елементи такі як left, right, bottom
   * **stick** - відносно фіксоване позиціонування, він залишається в потоці але коли ми задаємо якусь властивість top: 0; або інше значення він зафіксується коли буде доскролений до позицій яку ми вказали
   * **inherit** - успадкування батьківського значення
3. Describe z-index and how stacking context is formed ?

* Z-індекс - це властивість, яка дозволяє розробнику складати елементи в CSS. В основному це тривимірна властивість, тому вона дозволяє розробнику вибрати, наскільки близьким буде елемент. Так формується контекст укладання.

1. What’s the difference between inline-block, block, inline?

Які значення існують у властивості **display**

* **inherit** - успадковує властивість від батьківського елементу
* **block** - формує структурний блок, такий як тег div
* **inline** - дозволяє згенерувати рядковий елемент такий як тег **span**
* **inline-block** - дозволяє згенерувати рядково блочний елемент
* **flex** - дозволяє зробити адаптивний контейнер
* **grid** - дозволяє нам створити сітку, яка дозволяє нам визначити рядки та стовпці де будуть розміщуватися наші елементи

Compared to display: inline, the major difference is that **inline-block** allows to set a width and height on the element. Also, with display: inline, top and bottom margins & paddings are not respected, and with display: inline-block they are.

Now, the difference between display: inline-block and display: block is that, with **display: block**, a line break happens after the element, so a block element doesn’t sit next to other elements. Here are some visual examples:

1. Can you apply padding-top & padding-bottom to inline level element?

* Ні, простір зверху і знизу додати не можна, можна додати з права і ліва.

1. What is margin collapse?

* margin collapse трапляються, коли два вертикальних поля стикаються один з одним. Якщо одне поле більше іншого, тоді це поле перевизначає інше.

.module\_\_top {

margin-bottom: 25px;

background-color: #f38a6d;

}

.module\_\_bottom {

margin-top: 50px;

background-color: #3bbfef;

}

нижній колір їсть відступ вернього і буде лише 50 px.

1. What is box-sizing? Why we need it?

* box-sizing Визначає, як обчислюються ширина та висота елемента: чи повинні вони включати відступи та межі чи ні, дозволяє включити відступ і межу до загальної ширини та висоти елемента.

1. What’s the difference between display: none & visibility: hidden & opacity: 0? ([answer](https://htmlacademy.ru/blog/boost/frontend/short-12))

visibility: hidden - ця властивість приховає елемент, але займає простір.

display: none - він приховає елемент, і це дозволить іншим елементам зайняти простір.

opacity: 0 - ця властивість використовується, щоб зробити елемент прозорим, він займає простір.

1. What’s the visually hidden snippet?

* Це використання visibility: hidden в такому випадку елемент знаходиться на сторінці сусідні облоки його бачать але з ним не можна взаємодіяти користувачеві, зато можна зробити видимим в потрібний момент.

1. List all options how to center element horizontally and vertically? Inline and block elements. ([answer](https://css-tricks.com/centering-css-complete-guide/))

Наприклад:

Inline - H2 { text-align: center }

Вертикальне центрування:

div.container3 {

height: 10em;

**position: relative** } /\* 1 \*/

div.container3 p {

margin: 0;

**position: absolute;** /\* 2 \*/

**top: 50%;** /\* 3 \*/

**transform: translate(0, -50%)** }

div.container5 {

height: 10em;

**display: flex;**

**align-items: center** }

div.container5 p {

margin: 0 }

Вертикальне і горизонтальне центрування

div.container4 {

height: 10em;

position: relative }

div.container4 p {

margin: 0;

background: yellow;

position: absolute;

top: 50%;

**left: 50%;**

**margin-right: -50%;**

transform: translate(**-50%,** -50%) }

div.container6 {

height: 10em;

display: flex;

align-items: center;

**justify-content: center** }

div.container6 p {

margin: 0 }

CENTERING IN THE VIEWPORT

section {

background: black;

color: white;

border-radius: 1em;

padding: 1em;

**position: absolute;**

**top: 50%;**

**left: 50%;**

**margin-right: -50%;**

**transform: translate(-50%, -50%)** }

1. What is flexbox? ([article](https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_Flexible_Box_Layout/Basic_Concepts_of_Flexbox))([one more opinion](https://youtu.be/4OWOEJbqhHM?t=1785))

* Це управління елементами в ряду по горизонталі і вертикалі, а також перенос елементів

1. How can we change the order of elements in flexbox?

flex-direction:

* row
* row-reverse
* column
* column-reverse

1. How margin: auto works in flexbox?

Цей елемент автоматично розширить своє зазначене поле, щоб зайняти зайвий простір у гнучкому контейнері, залежно від напрямку, в якому застосовується автоматичне поле.

1. What does object-fit: cover and object-fit: contain mean?

У **object-fit** власності визначає , як елемент реагує на висоту і ширину до його вмісту в коробці.

* contain: збільшує або зменшує розмір зображення, щоб заповнити поле, зберігаючи пропорції.
* cover: зображення заповнює висоту та ширину вікна, знову зберігаючи пропорції, але часто обрізаючи зображення в процесі.

1. What’s the difference between vector and raster?

На відміну від растрової графіки, яка складається з кольорових пікселів, розташованих для відображення зображення, векторна графіка складається з контурів, кожен з яких має математичну формулу (вектор), яка вказує контуру, яку вона має форму і яким кольором вона облямована або заповнена .

1. Explain different kind of transforms?
2. Rotation: обертання предмета навколо нерухомої точки без зміни його розміру або форми
3. Translation: переміщення об’єкта в просторі без зміни його розміру, форми чи орієнтації
4. Dilation: розширення або скорочення об’єкта без зміни його форми чи орієнтації
5. Reflection: перегортання об’єкта через лінію без зміни його розміру чи форми
6. What is transition in css3? Explain with syntax?

Переходи CSS дозволяють плавно змінювати значення властивостей протягом заданої тривалості.

Приклад:

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s, height 4s;

}

div:hover {

width: 300px;

height: 300px;

}

1. What is animation in css3? Explain with syntax?

Анімація дозволяє елементу поступово переходити від одного стилю до іншого.

Ви можете змінити скільки завгодно властивостей CSS, скільки завгодно разів.

1. What are right properties for animations. Why transform & opacity are preferred? ([answer](https://developers.google.com/web/fundamentals/design-and-ux/animations/animations-and-performance?hl=ru))

Кращі властивості для анімації є transform & opacity, кращі вони тому що на їх реалізацію потрібно меньше ресурсів.

1. How to write efficient CSS? Explain common rules, technologies, methodologies. ([answer](https://www.freecodecamp.org/news/7-important-tips-for-writing-better-css/))

7 правил:

1. DRY "Don't Repeat Yourself"

наприклад для кнопок написати загальні правила один раз у додатковому класі, а різні правила - в інших класах

.button {

color: white;

border-radius: 5px;

padding: 10px 20px;

text-align: center;

font-size: 16px;

}

.primary-button {

background: blue;

}

.form-button {

background: green;

}

.cancel-button {

background: red;

}

1. Ім'я селектора має бути **самоописовим і читабельним**

.article-paragraph {

// Rules

}

.contact-form {

// Rules

}

**BEM** (модифікатор елемента блоку)

1. Не використовуйте вбудовані стилі
2. Уникати тегу **!important**; Це погано, оскільки правила CSS можуть відрізнятися від одного селектора до іншого, від батьківського селектора до дочірнього. Єдиний спосіб замінити **важливий тег** - використовувати **інший важливий тег** . І так до безкінечності, скоро код неможливо буде підтримувати.
3. Використовуйте препроцесор - ми можемо визначити змінні, функції та комбінації, ми можемо імпортувати та експортувати наші файли CSS в інші файли CSS і ми можемо писати вкладений код CSS

## Використовуйте скорочення

.article-container {

padding: 10px 15px 20px 15px;

margin: 10px 15px;

border: 1px solid black;

}

## Додайте коментарі, коли це необхідно, тільки якщо дійсно не зрозуміла частина

1. What are CSS methodologies?

* Методологія CSS - це набір керівних принципів для написання модульного, багаторазового та масштабованого коду.

1. Explain BEM methodology. ([article](https://ru.bem.info/methodology/quick-start/))

Це методологія яка дозволяє робити верстку яку легко підтримувати на великих і середніх проектах.

1. What is stylelint? When to use it?

* Stylelint - це потужний сучасний CSS-лінтер, який допомагає вам застосовувати послідовні правила та уникати помилок у таблицях стилів
* Використовувати завжди при написанні стилів

1. What are vendor prefixes?

префікси постачальників - це спосіб для вашого браузера підтримувати нові функції CSS, перш ніж вони стануть повністю підтримуваними у всіх браузерах.

# Preproccessors

1. What are preprocessors? Name few of them?

Препроцесори CSS - це мови сценаріїв, які розширюють можливості CSS за замовчуванням. Вони дозволяють нам використовувати логіку в коді CSS, таку як змінні, вкладеність, успадкування, комбінації, функції та математичні операції. ... Всі препроцесори CSS роблять подібні речі, але по-різному та зі своїми синтаксисами.

* SASS
* LESS

## Stylus

1. Explain sass main features (nesting, mixins, functions, variables, placeholders) ([answer](https://sass-lang.com/guide))

* Змінні Ви можете зберігати такі речі, як кольори, стеки шрифтів або будь-яке значення CSS. Сасс використовує $ символ, щоб зробити щось змінним.

$font-stack: Helvetica**,** sans-serif;

$primary-color: #333;

body {

**font**: 100% $font-stack;

**color**: $primary-color;

}

* Вкладання Sass дозволить вам вкласти ваші селектори CSS таким чином, щоб слідувати тій самій візуальній ієрархії вашого HTML.

## Модулі Ви можете створити часткові файли Sass, які містять невеликі фрагменти CSS, які ви можете включити в інші файли Sass. Це чудовий спосіб модулювати ваш CSS і допомогти полегшити обслуговування. Частково - це файл Sass, названий починається з подвійного підкресленням. Ви можете назвати це приблизно так \_partial.scss. Підкреслення дозволяє Сассу знати, що файл є лише модулемі що його не слід генерувати у файл CSS . Частинки Sass використовуються з @use правилом.

ось так:

// styles.scss

@use 'base';

.inverse {

background-color: base.$primary-color;

color: white;

}

## Міксіни

Деякі речі в CSS дещо нудно писати, особливо з CSS3 та багатьма префіксами постачальників, які існують. Комбінація дозволяє створювати групи декларацій CSS, які ви хочете повторно використовувати на своєму веб-сайті. Ви навіть можете передати значення, щоб зробити ваш мікс більш гнучким. Добре використовувати міксин для префіксів постачальників. Ось приклад для transform.

**@mixin** **transform**($property) {

-webkit-transform: $property;

-ms-transform: $property;

**transform**: $property;

}

**.box** { **@include** **transform**(**rotate**(30deg)); }

); }

Для створення міксину ви використовуєте @mixin директиву та даєте їй назву. Ми назвали наш міксин transform. Ми також використовуємо змінну $property всередині дужок, щоб ми могли передати все, що хочемо. Після створення міксину ви можете використовувати його як декларацію CSS, починаючи з @include назвою міксину.

## Розширення / спадкування Це одна з найкорисніших функцій Sass. Використання @extend дозволяє вам ділитися набором властивостей CSS від одного селектора до іншого це DRY

## Оператори Зарахування математики у вашому CSS дуже корисно. Sass має кілька стандартних математичних операторів , таких як +, -, \*, /і %.

1. Difference between sass and scss?

* SASS використовується, коли нам потрібен оригінальний синтаксис, синтаксис коду не потрібен для SCSS.
* SASS дотримується суворого відступу, SCSS не має суворого відступу.
* SASS має вільний синтаксис з пробілами і без крапок з комою, SCSS більше нагадує стиль CSS, а використання крапок з комою та фігурних дужок є обов’язковим.
* Розширення файлу SAAS - .sass, а розширення SCSS - .scss.
* SASS має більшу спільноту та підтримку розробників, ніж SCSS.

1. What is postcss?

PostCSS - це інструмент розробки програмного забезпечення, який використовує плагіни на основі JavaScript для автоматизації звичайних CSS-операцій.

# JavaScript

1. What is a global object?([article](https://javascript.info/global-object))

Глобальний об'єкт надає змінні і функції, доступні в будь-якому місці програми. За замовчуванням це ті, що вбудовані в мову або середу виконання.

У браузері він називається window.

1. What is scope? (Область видимости)([article](https://developer.mozilla.org/en-US/docs/Glossary/Scope))

Область видимості це - контекст, у якому значення та вирази "видимі" або на них можна посилатися. Область нащадок може мати доступ до області батька але не навпаки.

1. What is the difference between var, let and const?([article](https://javascript.info/var))

**var** - це старіший приклад оголошення змінних, мінуси її в тому що він створюється глобально і її можна перевизначити, тобто я можу другий раз створити змінну з таким іменем і присвоїти її нове значення. Їх область видимості обмежують лише функції в яких вони були створені.

**let** - змінна за допомогою ключового слова let видима тільки в блоці якому створена, крім того ми не можемо її заново перестворити, тобто створити змінну з таким самим іменем, коли ми доступаємося до змінної до її оголошення в нас виникне помилка.

**const** - змінна створена за допомогою ключового слова const має таку саму поведінку як let але її не можна переприсвоїти значення яке ми її вже задали

1. What’s the difference between function expression and function declaration? Which one to use?

Різниця між function declaration і function expression в наступному

перша відмінність

* синтаксис **function declaration** ключове слово function пишемо назву функції потім вказуємо аргументи якщо є і далі йде тіло функції
* **function expression** це оголошення функції в контексті будь якого виразу, тобто ми можемо нашу функцію присвоїти в змінну яку перед тим ми створили

друга відмінність в тому що функції які оголошенні як **function declaration** створюються інтератором до виконання кодуми можемо викликати в будь якому місці нашого коду

**function expression** можна викликати тільки після оголошення самої функції

В залежності від задачі.

1. What is lexical environment? (Лексическое окружение)([article](https://javascript.info/closure#lexical-environment))

"Лексичне середовище" це об'єкт складається з двох частин:

Запис навколишнього середовища - об'єкт, який зберігає всі локальні змінні як свої властивості (і деяку іншу інформацію, наприклад значення this).

Посилання на зовнішнє лексичне середовище , те, що пов'язане із зовнішнім кодом.

1. What is execution context? ([article](http://javascript.info/recursion#the-execution-context-and-stack))

Контекст виконання - спеціальна внутрішня структура даних, що містить інформацію про виклик функції. Вона включає в себе конкретне місце в коді, в якому знаходиться інтерпретатор, локальні змінні функції, значення **this** та іншу службову інформацію.

Контекст виконання складається з трьох етапів:

1. Запам'ятовується поточний контекст на вершині стеку.
2. Створюється новий контекст для вкладеного виклику.
3. Коли виконання вкладеного виклику закінчується - контекст попереднього виклику відновлюється, і виконання відповідної функції триває.
4. Explain closures.

**Замикання** - являє собою конструкцію коли функція створена в одній області видимості запам'ятовує своє лексичне оточення навіть у тому випадку коли вона виконується поза своєю областю видимості, три основних аспекти в замиканні

* це є зовнішня функція
* це є змінні створенні в зовнішнішні функції тобто які будуть записувати її лексичне оточення
* а також внутрішня функція яка обробляє дії над змінними які створені в зовнішні функції

**Рекурсія** це функція яка в своєму тілі викликає саму себе.

Рекурсія корисна в таких випадках коли завдання може бути природно поділене на аналогічні завдання просто в більш простіші формі.

**Обчислення факторіалу.**

1. What is this in JavaScript? How does it change its value? How does it work with arrow functions?

**this**  набуває різних значень в залежності від того в якому контекстів його буде викликано

* якщо в загальному скоупі то це буде об’єкт window
* якщо в середині функції то ця функція і стане **this**

1. Explain methods bind, call, apply.

Метод **bind** створює копію функції та встановлює ключове слово this, тоді як методи call і apply встановлюють this і негайно викликають функцію

Різниця між call() і apply() в тому що apply() метод приймає масив аргументів замість значень, розділених комами.

1. What is prototype? Explain how inheritance works in javascript?

**Prototype** - це механізм, за допомогою якого об'єкти JavaScript успадковують властивості один від одного.

встановлення наслідування Object.setPrototypeOf(child, father); властивостей

Встановлюємо від кого наслідує Object.getPrototypeOf(child)

властивість **proto** це - обчислювальна властивість яку ми отримуємо від батьківського об’єкту.

1. Explain async await and promises

**Promise** це об'єкт який відображає остаточне завершення (або неуспіх) асинхронної операції та значення, яке вона повертає.

**Async / Await** - це симпатичний синтаксис для обробки кількох **Promise** в синхронному коді. Коли ми ставимо ключове слово **async** перед оголошенням функції, воно поверне **Promise** , і ми можемо використовувати ключове слово **await** всередині нього, яке блокує код, доки **Promise**, який він чекає, буде успішним або відхилиться з якимось значенням.

1. What is event loop? Explain it. ([article](https://medium.com/devschacht/javascript-eventloop-explained-f2dcf84e36ee))

Інтерпретатор читає код, закидає функції в стек і по черзі викликає, якщо в функції є якийїсь вкладений call то він відпрацює після повного виконання stack. Тобто якщо у функції був setTimeout то колбек в ньому відпрацює після виконання основного stack функцій.

1. What JS data types do you know? How to check the variable type?

* number
* string
* boolean
* bigint
* symbol
* object

Перевірити можна за допомогою **typeOf**

для цих типів потрібні окремі перевірки:

* null
* undefined

1. What are strict and non-strict comparison in JS? What is the difference between them?

При не суворому порівняні два значення зводяться до одного типу даних і тоді порівнюються? але порівнюються тільки значення.

При суворому - порівнюються і тип даних кожного з значень.

1. What is the ternary operator? When is it a good idea to use it and when not?

Яко наш вираз може мати тільки два значення 0/1 тоді тернарник ідеальний варіант, код буде читабельним і гарно виглядатиме.

1. What logical operators do you know? What result do they produce?

* && - якщо є -зупиняється на неправді і поверне **false**
* || - якщо є -зупиняється на правді і поверне **true**
* ! - Дії!:

Спочатку наводить аргумент до логічного типу true / false.

Потім повертає протилежне значення.

1. What kinds of JS loops are you familiar with? In what scenarios is it best to use each of them?

* for - для ітерабельних об’єктів, коли знаєм необхідну кількість ітерацій
* for of - для перебору віх ітеруємих об’єктів
* for in - для перебору властивостей в об’єкті

**for of; for in** ([article](http://coldfox.ru/article/5c7ffe64bbf20e61c12c7348/%D0%9E%D1%82%D0%BB%D0%B8%D1%87%D0%B8%D0%B5-for-of-%D0%BE%D1%82-for-in-%D0%B2-javascript))

* while - якщо не знаємо необхідну кількість ітерацій
* do while - якщо потрібно спочатку виконати тіло функції, а тоді перевірити умову

1. Name a few ways to round a number in JavaScript.
2. How to convert a numeral value into a string and vice versa? When can such a conversion come in handy?
3. How can the characters of a string be iterated through in JavaScript?
4. What is the best way to check whether a string contains a given substring in JavaScript? And what if the position in the string is required?
5. How can both leading and trailing whitespace be removed from a string? How would you implement prepending a few spaces to a string so that the string becomes of a given length?
6. What is an object? How to access properties of an object?
7. How can one check whether an object has a certain property?
8. How can a property be removed from an object?
9. How can the properties of an object be iterated through?
10. What are getters and setters?
11. What is an array? How can the elements of an array be iterated through?
12. What are some methods for adding and removing elements from an array that you remember? How do they work?
13. What are some methods for searching within an array that you remember? How do they work?
14. How do the following array methods work: filter, map, some, every, reduce?
15. How would you sort an array by a custom characteristic of its elements?
16. What is destructuring assignment? Give a few examples.
17. What is rest syntax? What is spread syntax? What is the difference between them?
18. What is the Set data structure for? What methods does it implement?
19. What is the Map data structure for? What methods does it implement?
20. What are WeakSet and WeakMap and how do they differ from their “strong” counterparts?
21. What is the application of regular expressions? What are some basic ways to work with them in JavaScript?
22. What is the meaning of parentheses, brackets, and braces in regular expressions?
23. What is the meaning of the following characters in regular expressions: +, \*, ?, ^, $?
24. What is recursion? What is a stack?
25. What is a class?
26. What are static properties of a class?
27. How would you check whether an object belongs to a certain class?
28. What are the means of error handling in JavaScript?
29. Why do we need modularity in our applications? What JS module loading tools are you familiar with?
30. Describe the flow of downloading data from a remote server in JavaScript. What JS data fetching tools are you familiar with?
31. Explain event bubbling?([article](https://javascript.info/bubbling-and-capturing))
32. What is event delegation?([article](https://javascript.info/event-delegation))

# Promises

* Що таке Promise?
* Який стан буває в Promise? Як його змінити?
* Які методи є в Promise?(then, catch, finally)
* Які статичні методи є в Promise?(Promise.all, Promise.race, Promise.resolve, Promise.reject, Promise.allSettled)
* Для чого Promise.all, як працює, синтаксис?
* Як створити Promise?
* Що повертають методи then, catch?

# Fetch

* Що таке fetch?
* Які аргументи приймає fetch?
* Що повертає fetch?(Що всередині цього промісу?)
* Які властивості і методи Response ти знаєш?
* Як по-іншому можна зробити HTTP запит?

# TypeScript

1. What is TypeScript and what is its purpose? ([article](https://en.wikipedia.org/wiki/TypeScript))
2. How to work with arrays in TypeScript? ([docs 1](https://www.typescriptlang.org/docs/handbook/basic-types.html#array), [docs 2](https://www.typescriptlang.org/docs/handbook/basic-types.html#tuple))
3. Hot to work with objects in TypeScript? What is interface? ([docs 1](https://www.typescriptlang.org/docs/handbook/basic-types.html#object), [docs 2](https://www.typescriptlang.org/docs/handbook/interfaces.html))
4. What is enum and how is it implemented in TypeScript? ([docs](https://www.typescriptlang.org/docs/handbook/basic-types.html#enum))
5. What are union types? ([docs](https://www.typescriptlang.org/docs/handbook/advanced-types.html#union-types))
6. What are the universal and impossible types? ([docs 1](https://www.typescriptlang.org/docs/handbook/basic-types.html#any), [docs 2](https://www.typescriptlang.org/docs/handbook/basic-types.html#never))
7. What are explicit and inferred types? ([docs](https://www.typescriptlang.org/docs/handbook/type-inference.html))

# Web

1. What’s the difference between relative and absolute path?
2. What are screenreaders?
3. How to make the page working for people with disabilities?
4. What are cookies, localStorage and sessionStorage. What is the difference between them?
5. What does http mean?

*http* - це протокол для веб-сторінок, в цьому протоколі дані передаються у відкритому виді, а в протоколі *https* у зашифрованому виді.

1. Explain in maximum details what modern browser does when user types url and hits enter. ([article](https://medium.com/@maneesha.wijesinghe1/what-happens-when-you-type-an-url-in-the-browser-and-press-enter-bb0aa2449c1a))
2. Explain page rendering process (calc styles => reflow => repaint => composition) ([article](https://medium.com/@mustafa.abdalmogoud/how-the-browser-renders-html-css-27920d8ccaa6)) ([article RU (2019)](https://habr.com/ru/company/yandex/blog/468165/))
3. What is critical css?
4. How linked CSS files affect the rendering of the page? ([article](https://developers.google.com/web/fundamentals/performance/critical-rendering-path/render-blocking-css))
5. What is the difference between including <script> in <head> and before <body> end?
6. What does async and defer mean in <script>?
7. What is the best way to include CSS and JS in HTML page?
8. How to improve website’s performance?
9. What is shadow DOM?
10. What is REST API? How is it built?
11. What is the difference between PUT and PATCH?
12. Explain CORS. Why does it exist?
13. Explain OSI model ([article](https://en.wikipedia.org/wiki/OSI_model))

# General

* Explain git flow.
* What resources do you use to improve your knowledge about web programming?
* How long have you been studying programming?
* Do you have a github account?
* How many web projects do you have?

# React interview questions

1. Що таке React і в чому полягають його переваги?

React — це JavaScript-бібліотека для створення інтерфейсів користувача.

**React** дозволяє розробникам створювати великі веб-застосунки, які використовують дані, котрі змінюються з часом, без перезавантаження сторінки. **Його** мета полягає в тому, щоб бути швидким, простим, масштабованим. **React** обробляє тільки користувацький інтерфейс у застосунках.

1. Чи можна до вже існуючого сайту підключити лише один новий компонент на React? Як?

так можна

1. Що таке JSX? Який у нього синтаксис?

JSX - це розширення синтаксису для JavaScript. Рекомендується використовувати його в React, щоб описати, як повинен виглядати інтерфейс користувача. JSX може нагадувати мову шаблонів, але з усіма перевагами JavaScript.

1. Чи обов’язково використовувати JSX із Реактом?

JSX не є вимогою для роботи з React. Використання React без JSX є найзручнішим тоді, коли ви не бажаєте налаштовувати компіляцію у вашому середовищі збірки.Кожен JSX-елемент являє собою “синтаксичний цукор” для виклику React.createElement(component, props, ...children). Отже, все що можна зробити за допомогою JSX, може також бути виконаним на чистому JavaScript.

1. Що таке Virtual DOM?

Віртуальний DOM (VDOM) - це концепція програмування, в якій ідеальне чи «віртуальне» представлення інтерфейсу користувача зберігається в пам’яті і синхронізується зі «справжнім» DOM за допомогою бібліотеки, такої як ReactDOM. Цей процес називається [узгодженням](https://uk.reactjs.org/docs/reconciliation.html).

Virtual DOM — інструмент, що дозволяє взаємодіяти з елементами DOM простіше та ефективніше. Virtual DOM представлено у вигляді об'єкта Javascript, який ми можемо змінювати так часто, як нам потрібно. Зміни, здійснені над об'єктом накопичуються, а фактичний DOM оновлюється направлено та рідшe.

1. Що таке компонент і в чому різниця між компонентом-функцією та компонентом-класом?

У чому відмінність між компонентом та JSX-елементом?

Компоненти дозволяють розділити інтерфейс користувача на незалежні частини, придатні до повторного використання, і сприймати їх як такі, що функціонують окремо один від одного.

Концептуально компоненти є подібними до функцій JavaScript. Вони приймають довільні вхідні дані (так звані “пропси”) і повертають React-елементи, що описують те, що повинно з’явитися на екрані.

1. Що таке props? Як отримати до них доступ і як їх поміняти? ([example](https://youtu.be/xMZ2ogz-Q-s?t=344))

React-компонент приймає єдиний аргумент “пропс” (скорочено від *properties* - властивості), який є об’єктом з даними і повертає React-елемент.

props - це аргументи для функціонального або класового компоненту.

1. Що таке state? Як отримати до нього доступ і як його поміняти? ([example](https://youtu.be/xMZ2ogz-Q-s))

Стан подібний до пропсів, але він приватний і повністю контролюється компонентом.

1. Які є методи життєвого циклу React-компонента? ([article](https://programmingwithmosh.com/javascript/react-lifecycle-methods/))

**Mounting (**render()**) - рендер компоненту**

* **componentDidMount ()** - На відміну від методу **render ()** ,  **componentDidMount ()** дозволяє використовувати **setState ()** . Виклик setState () тут оновить стан і спричинить інший візуалізацію, але це відбудеться до того, як браузер оновить інтерфейс. Це робиться для того, щоб користувач не бачив жодних оновлень інтерфейсу при подвійні візуалізації.

**Update**

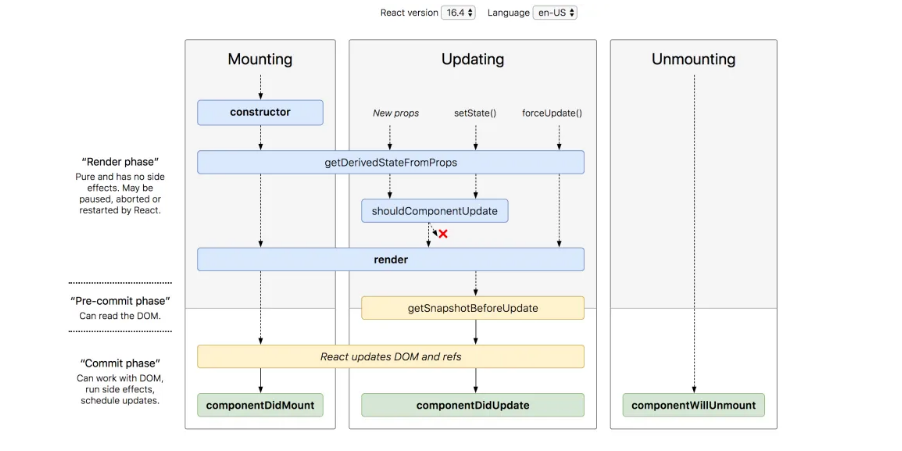
* **componentDidUpdate () -** Цей метод життєвого циклу викликається, як тільки відбувається оновлення, **state** або **props**.

**Unmount**

* **componentWillUnmount ()** - цей метод життєвого циклу викликається безпосередньо перед тим, як компонент демонтується та знищується. Якщо вам потрібно виконати якісь дії з очищення, це буде правильне місце.

Загальні заходи очищення, що виконуються цим методом, включають очищення таймерів, скасування викликів API або очищення кеш-пам’яті.

* **shouldComponentUpdate ()** - використовується, щоб повідомити React, якщо на компонент не впливають зміни стану та властивостей.



1. Як можна навісити подію на JSX-компонент? Чи можна навішувати події на компоненти?

Ні на компоненти події навішувати не можна, так як це об’єкт а не елемент, і така подія як наприклад onClick - передасться в компонент як props, і не працюватиме як атрибут до елементу.

1. Результат яких типів може повертати метод render (або компонент-функція)?

React-елементи, Масиви та фрагменти, Портали, Рядки і числа, Логічні значення чи null.

1. У яких випадках потрібно присвоювати елементам та/або компонентам атрибут key?

Яке значення слід присвоювати цьому атрибуту? Для чого він потрібен?

“Ключ” - це спеціальний рядковий атрибут, який потрібно вказувати при створенні списку елементів.Ключі допомагають React визначити, які елементи були змінені, додані або видалені. Ключі повинні бути надані елементам всередині масиву, щоб надати елементам стабільну ідентичність.Найкращий спосіб вибрати ключ - це використати рядок, котрий буде вочевидь відрізняти елемент списку від його сусідів.

1. Що таке контрольовані компоненти? ([article](https://reactjs.org/docs/forms.html#controlled-components))

React-компонент, який буде рендерити форму, також буде контролювати її поведінку у відповідь на введення даних користувача. Значення елемента форми в цьому випадку буде контролювати React, а сам елемент буде називатися “керованим компонентом”.

Якщо ми в атрибут **value** присвоюємо якесь значення з **state** для реакту це буде контрольований компонент.

1. Чи відрізняються чимось атрибути та події для полів форми у Реакті від звичайних у DOM?

У React усі властивості й атрибути DOM (включно з обробниками подій) мають використовувати *camelCase* нотацію.

Так, тим що:

* <input type="text">, <textarea> і <select> всі вони **приймають атрибут value**, який можна використовувати для реалізації керованого компоненту.

поля введення будуть контрольовані і оновлюватися тільки через викик setState()

* select - може в **value** прийняти масив значень

<select multiple={true} value={['B', 'C']}>

1. Що таке неконтрольовані компоненти? У яких випадках є сенс їх використовувати?([article](https://reactjs.org/docs/uncontrolled-components.html))

Це компоненти які контролюються DOM і оновлюються без участі React.

В більшості випадків в React потрібно використовувати контрольовані компоненти, виключенням може бути лише тег - <input type="file" /> у React завжди є неконтрольованим компонентом, оскільки його значення може встановити лише користувач, а не програмно.

1. Що таке підйом стану? Наведіть приклад ситуації, коли ним варто скористатися. ([article](https://reactjs.org/docs/lifting-state-up.html))

Обмін станом у React здійснюється шляхом переміщення його до найближчого спільного предка компонентів, які його потребують. Це називається “підйом стану вгору”.

Як це працює на практиці:

Маємо компонент input з контрольованим введенням переносив хук який записує value з input до потрібного батьківського компоненту, тоді саму функцію і її значення переда'мо всім компонентами які її потребують в пропсах. При зміні значення value батьківський і дочірні компоненти перемалюються.

1. Що таке композиція компонентів? ([article](https://reactjs.org/docs/composition-vs-inheritance.html))

Розбивка великого компоненту на менші і вкладання один в одного в якості props.children

function FancyBorder(props) {

return (

<div className={'FancyBorder FancyBorder-' + props.color}>

{props.children}

</div>

);

}

Це дозволяє іншим компонентам передавати довільні дочірні елементи до них шляхом вкладання JSX:

function WelcomeDialog() {

return (

<FancyBorder color="blue">

<h1 className="Dialog-title">

Welcome

</h1>

<p className="Dialog-message">

Thank you for visiting our spacecraft!

</p>

</FancyBorder>

);

}

В даному прикладі в нас є два компоненти перший дозволяє зробити нам border-color який прийде з props разом з JSX розміткою.

1. Що таке React.Fragment і в яких випадках ними варто користуватися? ([article](https://reactjs.org/docs/fragments.html))

Повернення кількох елементів з компонента є поширеною практикою в React. Фрагменти дозволяють формувати список дочірніх елементів, не створюючи зайвих вузлів в DOM.

1. Для чого потрібен PropTypes і які види перевірок у ньому доступні? ([article](https://reactjs.org/docs/typechecking-with-proptypes.html))

PropTypes - потрібен для перевірки типів даних які приходять з props-ів щоб зменшити кількість можливих помилок при розробці.

Типи:

* optionalBool: PropTypes.bool,
* optionalFunc: PropTypes.func,
* optionalNumber: PropTypes.number,
* optionalString: PropTypes.string,
* optionalSymbol: PropTypes.symbol,
* optionalObject: PropTypes.object,
* optionalObjectOf: PropTypes.objectOf(PropTypes.number)
* optionalArray: PropTypes.array,
* optionalArrayOf: PropTypes.arrayOf(PropTypes.number),

# Advanced React

1. Що таке ref? У яких випадках ними варто користуватися і як? ([article](https://uk.reactjs.org/docs/refs-and-the-dom.html))

Рефи надають доступ до DOM-вузлів чи React-елементів, що створюються під час рендеру.

Коли використовувати рефи:

* Контроль фокусу, виділення тексту чи контроль програвання медіа.
* Виклик імперативної анімації.
* Інтеграція зі сторонніми DOM-бібліотеками.

Уникайте використання рефів для будь-чого, що можна зробити декларативно.

Приклад:

class MyComponent extends React.Component {

constructor(props) {

super(props);

**this.myRef** = React.createRef();

}

render() {

return <div ref=**{this.myRef**} />;

}

}

Особливість **ref** вони не перемальовують компонент.

1. Що за другий аргумент методу this.setState?

в аргументи отримує колбек який отримує два аргументи перший це **state** і попереднє значення **props**.

1. Що таке властивість children? ([article](https://senior.ua/articles/zanuryumosya-v-robotu-z-children-na-react))

props.children може бути будь-яким типом даних, наприклад таким як масив, функція, об'єкт і тп. Загалом, це може бути всім чим завгодно.

Це всі властивості і елементи які вкладені в цей об'єкт

1. Розкажи про паттерн render props.
2. Що таке HOC? Навіщо він потрібен? Чи може HOC мати власний state?
3. Що таке React.Context? Як і в яких випадках ним користуватися?
4. Що таке імутабельність? Який benefit вона дає в React?

Імутабельний/незмінний (англ. immutable) називається об'єкт, стан якого не може бути змінено після створення. Результат будь-якої модифікації такого об'єкта завжди буде новим об'єктом, при цьому старий об'єкт не зміниться.

Завдяки цій властивості реакт може порівнювати попередній об’єкт з поточним і розуміти чи потрібно перемальовувати сторінку.

1. Як можна оптимізувати рендеринг компонентів? (keys, shouldComponentUpdate, PureComponent, React.memo)
2. Які особливості static getDerivedStateFromProps?
3. Які є способи роботи з CSS в React? Плюси / мінуси
4. Переваги фукціональних компонентів над повноцінними (класовими ?) компонентами
5. Що таке server side rendering ([one more opinion](https://youtu.be/0iL7Ji5Ya8A?t=1415))

Це коли перша сторінка рендириться на стороні серверу (Node.js) і віддається вже готовою при першому запиті клієнту. Після чого вже в включається React і робить порівняння і локальні зміни при потребі.

1. Завдяки чому може бути перемалювання компоненту

* зміні state
* зміні пропсів
* перемалюваня батьківського компоненту

# Hooks

1. Що таке React hooks? В яких компонентах їх можна використовувати?

Хуки — це функції, за допомогою яких ви можете “зачепитися” за стан та методи життєвого циклу React з функціональних компонентів. Хуки не працюють всередині класів — вони дають вам можливість використовувати React без класів.

1. Перерахуй hooks які ти знаєш?

### **useState - Повертає значення стану та функцію, що оновлює його.**

### **useEffect - Приймає функцію, що містить імперативний, можливо з ефектами, код.**

### **useContext - Приймає об’єкт контексту (значення, повернуте з React.createContext) і повертає поточне значення контексту для нього. Поточне значення контексту визначається пропом value найближчого <MyContext.Provider>, що знаходиться вище у дереві компонентів.**

### **useReducer - Є альтернативою** [**useState**](https://uk.reactjs.org/docs/hooks-reference.html#usestate)**. Приймає редюсер типу (state, action) => newState і повертає поточний стан у парі з методом dispatch.**

### **useCallback - Повертає** [**мемоізовану**](https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D0%BC%D0%BE%D1%96%D0%B7%D0%B0%D1%86%D1%96%D1%8F) **функцію зворотнього виклику, котра змінюється лише тоді, коли одна з її залежностей змінюється.**

### **useMemo - Повертає** [**мемоізоване**](https://en.wikipedia.org/wiki/Memoization) **значення.useMemo повторно обчислить мемоізоване значення лише при зміні однієї з залежностей. Така оптимізація допомагає уникнути вартісних обчислень при кожному рендері.**

### **useRef - поверне змінний об’єкт рефу, властивість .current якого ініціалізується переданим аргументом (initialValue). Повернутий об’єкт буде зберігатись протягом всього часу життя компонента.**

### **useLayoutEffect - Сигнатура хука є ідентичною useEffect, але він запускається синхронно після всіх змін DOM. Використовуйте це для читання розмітки з DOM і синхронних повторних рендерів. Оновлення, заплановані всередині useLayoutEffect, будуть виконані синхронно перед тим, як браузер відобразить їх.**

### **useDebugValue - useDebugValue може бути використаний для відображення мітки користувацьких хуків у інструментах розробника React.**

1. Що таке Rules of hooks? Розкажи про них.([answer](https://uk.reactjs.org/docs/hooks-rules.html))

Rules:

### [**Використовуйте хуки тільки на вищому рівні**](https://uk.reactjs.org/docs/hooks-rules.html#only-call-hooks-at-the-top-level)

### [**Викликайте хуки лише з React-функцій**](https://uk.reactjs.org/docs/hooks-rules.html#only-call-hooks-from-react-functions)

Хуки — це функції JavaScript, але вони накладають два додаткових правила:

* Хуки слід викликати тільки на верхньому рівні. Не викликайте хуки всередині циклів, умов або вкладених функцій.
* Хуки слід викликати тільки з функціональних React-компонентів. Не викликайте хуки із звичайних функцій JavaScript. (Є тільки один виняток, звідки можна викликати хуки — це ваші власні хуки.)

1. **Який хук можна використати, як аналог this.state React.Component?.**
2. **Опиши синтаксис(initialValue, lazy initialvalue, return tuple), в чому відмінності від state в класах?(Doesn’t merge state object)**
3. Який хук заміняє методи життєвого циклу? Опиши синтаксис.

componentDidMount, componentDidUpdate, componentWillUnmount:

Хук [useEffect](https://uk.reactjs.org/docs/hooks-reference.html#useeffect) може замінити всі їхні комбінації (включно з [менш](https://uk.reactjs.org/docs/hooks-faq.html#can-i-skip-an-effect-on-updates) [частими](https://uk.reactjs.org/docs/hooks-faq.html#can-i-run-an-effect-only-on-updates) випадками).

1. Як реалізувати componentDidMount на хуках?

Потрібно вкликати хук **useEffect**(

першим параметром передати колбек в тілі якого виконати необхідну функцію,

і не передавати нічого другим параметром

)

1. Як реалізувати componentDidUpdate на хуках?

Потрібно вкликати хук **useEffect**(

першим параметром передати колбек в тілі якого виконати необхідну функцію,

другим параметром передати масив залежностей для оновлення компоненту

)

1. Як реалізувати componentWillUnmount на хуках?

Потрібно вкликати хук **useEffect**(

першим параметром передати колбек в тілі якого повернути ще одну функцію яка і буде спрацьовувати перед демонтажем компоненту

)

1. Які хуки для оптимізації ти знаєш?(useMemo, useCallback).

**useMemo** відрізняється тим, що він повертає не саму функцію, а результат її виконання.

**Використовується** useMemo для збереження результатів важких обчислень, наприклад обробка масиву.

**useCallback** отримує функцію і масив аргументів, і повертає одну й ту ж саму функцію, до тих пір, поки аргументи не змінилися.

**Використовується** useCallback, коли важлива постійність посилань на функцію. Наприклад, коли ми передаємо посилання в компонент, який використовує React.PureComponent або React.memo, або, коли функція використовується в якості аргументу в інших хуках

1. useMemo - синтаксис, переваги, відмінності від useCallback

Синтаксис однаковий:

* useMemo(() => computeExpensiveValue(b), [a]);
* useCallback(() => computeExpensiveValue(b), []);

Відміність в тому що useMemo повертає дані, а useCallback повертає посилання на туж функцію.

Переваги оптимізація роботи програми.

1. Поясни як працює конструкція(як React розуміє, що це 2 різних states)([answer](https://uk.reactjs.org/docs/hooks-rules.html))

const [name, setName] = useState('');

const [count, setCount] = useState(0);

**React** покладається на послідовність викликів хуків. Наш приклад працює тому, що послідовність викликів хуків є сталою для кожного рендеру. Доки послідовність викликів хуків залишається сталою між рендерами, React може співвідносити локальний стан між кожним з них.

# Redux

1. Що таке Redux ?

**Redux** - це бібліотека для глобального управління станом додатку, що дозволяє передавати дані без необхідності проходити реквізит через кожен рівень (так зване буріння реквізиту).

1. Якими є основні засади Редакса і в чому полягають його переваги?

* Весь стан додатку зберігається в одному місці
* Стан тільки для читання, на пряму змінити не можна
* Мутації написані, як чисті функції
* Додаток будується по односпрямованому потоку даних (<http://redux.js.org/docs/basics/DataFlow.html>). Односпрямований потік кардинально відрізняється від MVC, де C-V і C-M зазвичай двонаправлені. Односпрямований потік спрощує розуміння програми.

1. Які переваги дає redux?

* Завдяки інверсії контролю, згаданої вище, зникає необхідність оновлювати призначений для користувача інтерфейс, як тільки змінюється імплементація зміни станів. Додавати такі складні функції, як логування, скасування дії або навіть time-travel debugging, стає простіше простого. Інтеграційні тести зводяться лише до того, щоб перевірити, відправляється чи правильна дія, а для всього іншого досить юніт-тестів.
* Стан компонентів в React занадто громіздка для роботи з наскрізною функціональністю, яка зачіпає багато модулі програми, як, наприклад, інформація про користувача або оповіщення. Якраз для цього в Redux є дерево станів, незалежне від призначеного для користувача інтерфейсу. До того ж, при обробці стану поза інтерфейсу легше підтримувати стабільність, адже сериализация в localStorage або URL проводиться в єдиному місці.
* Редюсери дають більшу свободу в роботі з діями, які можна поєднувати, відправляти одночасно і навіть обробляти в стилі method\_missing.

1. Які слабкі місця ти бачиш в цьому підході?

* Actions можна розглядати як складні переходи між станами, але в основному вони лише задають одне значення. У додатках, зроблених на Redux, часто накопичується безліч таких простих actions, і це явно нагадує Java з написанням функцій сеттерів вручну.
* Один і той же фрагмент стану можна було б використовувати по всьому додатком, але в більшості випадків він відноситься до якоїсь однієї частини інтерфейсу. Перенесення стану з компонентів в сховище Redux - це просто додатковий перенаправлення без належного рівня абстракції.
* Функції-редюсери могли б впоратися з самими мудрими завданнями метапрограмування, але зазвичай зводяться до примітивної диспетчеризації action згідно його типу. Це не проблема для таких мов, як Elm або Erlang, які відрізняються лаконічним і виразним синтаксисом pattern matching, але в Javascript доводиться мати справу з громіздкими конструкціями switch.

Слабкі місця, написання більше коду.

1. Що таке стан у Редаксі?

Це місце де зберігається весь стан компоненту всі дані.

1. Що таке pure function?

Чистий функція - це функція, яка завжди для одного і того ж вхідного значення повертає однаковий результат.

1. **Які елементи стану можна було б виділити для базового функціоналу Google: введення пошукового запиту — отримання результатів — введення нового запиту — отримання нових результатів і т. д.?**
2. Що таке дії (actions)? Який у загальному випадку вони мають вигляд?

- Action - це прості об'єкти Javascript з властивістю type, що вказують, який тип дії виконується.

Action відправляються в центральне сховище за допомогою **store.dispatch ()**

**ВИГЛЯД**: const action = (item: number) => {type:TYPE\_ACTION, item }

1. Що таке action creators?

Це функції які повертають об'єкт з типом дії а також можуть отримувати додаткові дані для запису в store.

1. Чи обов’язково різні типи дій задавати константами?

Так обов’язково щоб не можна було створити двох однакових Action.

1. Що таке reducer? Які обов’язкові вимоги до них існують?

Reducers - це функції, які приймають **Action** і поточний стан і повертають результуючий стан.

Reducers повинні бути чистими функціями і не мати побічних ефектів.

Reducers стан тут не змінюється безпосередньо. Замість цього повертається новий стан, яким Редакс вже сам замінює старий.

1. Чому не можна змінювати окремі елементи стану, а треба оновлювати весь об’єкт стану?

Тому що в Redux немає можливості змінювати окремі елементи стану можна тільки оновити об’єкт повністю.

1. Як написати код у редюсері, що додаватиме до масиву items у стані рядок 'new item'?

const reducer = (*state* = initialState, *action*: *AnyAction*) => {

switch (*action*.type) {

case ADD\_NEW\_ITEM:

return {...*state*, item: [...item, *action*.item]};

default:

return *state*;

}

};

1. Що таке сховище (store)?

Сховище є вашим контейнером, а стан(state) живе в контейнері. Сховище — об'єкт з API, який дає змогу взаємодіяти з вашим станом, змінюючи його, запитуючи його значення тощо.

Store - це центральний об'єкт, який підтримує й оновлює стан додатку.

1. Які в нього є методи і події?

#### Store.subscribe()

Метод subscrbe дозволить передати функцію підписника в Сховище, через яке, після зміни дерева стану, можна передати нові зміни стану як аргумент в колбек .subscribe ().

#### Store.value

Властивість value буде налаштована як гетер (getter) і вона поверне дерево внутрішнього стану (щоб ми могли отримати доступ до властивостей).

Store також займається підпискою і скасуванням піписки слухачів.

1. Які є два способи задати початковий стан у Redux? ([answer](https://codeguida.com/post/1172))
2. initialState
3. ?
4. Як саме з технічного погляду працює спосіб задати початковий стан у коді редюсера?

Створити об’єкт initialState і задати в ньому початкові дані.

export const initialState = {

data: [],

loaded: false,

loading: false,

};

Далі у функції редюсера потрібно вказати аргумент state із значенням по замовчуванню initialState. Це налаштовує редюсер для початкового завантаження, коли ми викликаємо його у Сховищі, щоб зв'язати усі початкові стани у всіх редюсерах.

export function todosReducer(

state = initialState,

action: { type: string, payload: any }

) {

switch (action.type) {

case 'ADD\_TODO': {

const todo = action.payload;

const data = [...state.data, todo];

return {

...state,

data,

};

}

}

return state;

}

Якщо ви хочете використовувати синтаксис Store(reducers, initialState) , щоб передати початковий стан для всього Сховища, потрібно застосувати метод reduce на етапі створення Сховища.

export class Store {

private state: { [key: string]: any };

private reducers: { [key: string]: Function };

constructor(reducers = {}, initialState = {}) {

this.reducers = reducers;

this.state = this.reduce(initialState, {});

}

// ...

}

1. Як об’єднати два редюсери, що опікуються різними ділянками стану, в один?

Допоміжна функція **combineReducers** перетворює об'єкт, значеннями якого є різні функції редюсери, в одну функцію редюсер, яку можна передати в метод createStore.

Розміщуються в файлі **index** в папці **store** виглядають так:

const rootReducer = **combineReducers**({

posts: postsReducer,

detailsPost: detailsPostsReducer,

});

1. Що таке презентаційні компоненти та компоненти-обгортки? ([answer](https://medium.com/@dan_abramov/smart-and-dumb-components-7ca2f9a7c7d0), [an2](https://habr.com/ru/company/ruvds/blog/423157/))
2. **Презентаційні компоненти**. Вони відповідають лише за зовнішній вигляд програми, але не обізнані про стан Redux. Вони отримують дані через властивості і можуть викликати колбеки, які також передаються їм через властивості.
3. **Компоненти-контейнери**. Вони відповідальні за роботу внутрішніх механізмів додатки і взаємодіють зі станом Redux. Їх часто створюють з використанням react-redux, вони можуть здійснювати диспетчеризацію дій Redux. Крім того, вони підписуються на зміни стану.
4. Які параметри приймає та що повертає функція connect? (Все про [Connect](https://habr.com/ru/company/ruvds/blog/423157/))
5. Параметри:

connect ([mapStateToProps], [mapDispatchToProps], [mergeProps], [options]) - дозволяє створювати компоненти вищого порядку. Це потрібно для створення компонентів-контейнерів на основі базових компонентів React.

API react-redux connect () використовується для створення компонентів-контейнерів, які підключені до сховища Redux

1. connect () Повертає функцію вищого порядку. Виглядає так:

import React from 'react';

import {connect} from 'react-redux';

import Profile from './components/Profile';

function ProfileContainer (props) {

return (

props.loggedIn

? <Profile profile = {props.profile} />

: <Div> Please login to view profile. </ Div>

)

}

const mapStateToProps = function (state) {

return {

profile: state.user.profile,

loggedIn: state.auth.loggedIn

}

}

export default **connect** (mapStateToProps) (ProfileContainer);

1. Яке призначення функцій mapStateToProps та mapDispatchToProps і в чому полягає різниця між ними? ([answer](https://habr.com/ru/company/ruvds/blog/423157/))
2. Як передати параметр із props, що передали компоненту-обгортці, презентаційному компоненту, який він огортає?
3. Як користуватися провайдером з react-redux? Чому він потрібен?

<**Provider** store> - потрібно обгорнути весь додаток, це дозволяє робити стан Redux доступним для всіх компонентів-контейнерів в його ієрархії.

1. Де б ти викликав асинхроний запит до сервера через redux?

В Thunk тому що ця функція призначена для обробки таких запитів.

1. Що таке middleware в redux?

**Middleware -** знаходиться між кодом, який dispatch екшени (тобто генерує події) і редьюсером (тобто функцією, яка їх перетворює в новий стан), пропускає через себе всі виникаючі екшени, змінює їх (якщо вважає за потрібне) і відправляє далі - або не надсилає .

Особливість **middleware** в тому, що він не зобов'язаний бути чистою функцією - і тому він може робити сайд-ефекти на зразок запитів на сервер або чого завгодно.

1. Які middleware ти використовував на проектах?

* thunk

const store = createStore(

rootReducer,

composeWithDevTools(**applyMiddleware**(thunk)),

);

1. Що таке thunk? Як його писати і підключати?

**Thunk** - потрібні нам для роботи з асинхронними операціями **Promise** , і виконання додаткових дій перед відправкою даних в store**.**

**index.ts**

**import { createStore, combineReducers, applyMiddleware } from 'redux';**

**import { composeWithDevTools } from 'redux-devtools-extension';**

**import thunk from 'redux-thunk';**

**import postsReducer from './postsReducer';**

**const rootReducer = combineReducers({**

**posts: postsReducer,**

**});**

**// We automatically get types returned by concrete reducers**

**export type *RootState* = *ReturnType*<typeof *rootReducer*>;**

**const store = createStore(**

**rootReducer,**

**composeWithDevTools(applyMiddleware(thunk)),**

**);**

**export default store;**

Що таке локальні стилі в React ? Для чого їх використовують ?

Ref - не викликають перерендеригу