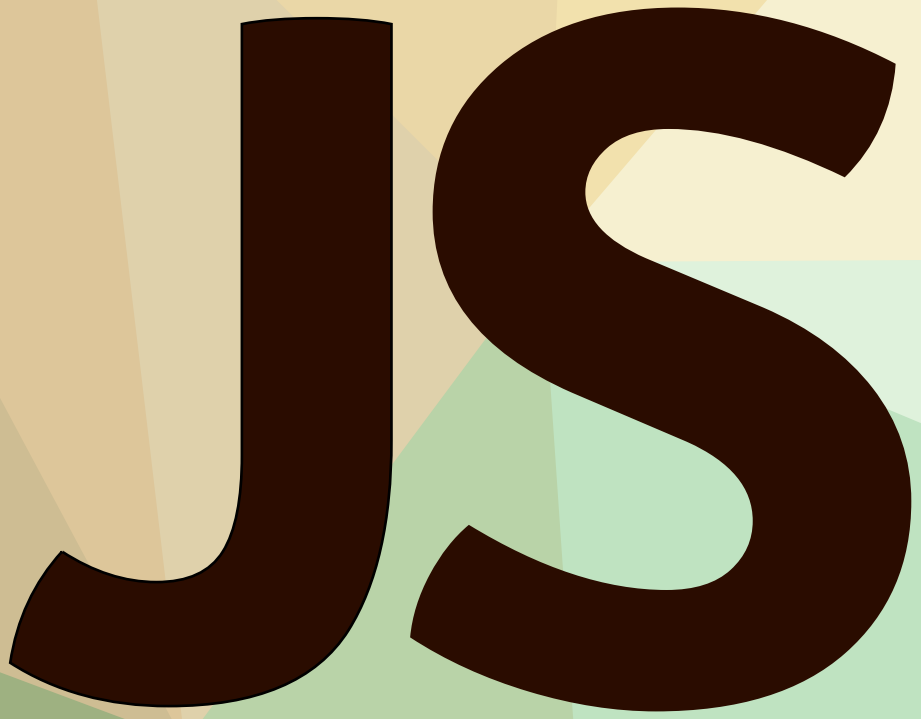
Частина 1

Мова програмування JavaScript



Ilya Kantor

Побудовано на 5 січня 2024 р.

Актуальна версія підручника знаходиться за адресою https://uk.javascript.info.

Ми постійно працюємо над покращенням підручника. Якщо ви знайдете якісь помилки, пишіть на нашому github.

- Вступ
  - Вступ до JavaScript
  - Довідники й специфікації
  - Редактори коду
  - Інструменти розробника

Тут ви можете вивчити JavaScript, починаючи з нуля і закінчуючи просунутими концепціями, як ООП.

Ми зосередимось на самій мові, зрідка роблячи примітки щодо середовищ її виконання.

Вступ

Про мову JavaScript і робоче середовище для розробки.

Вступ до JavaScript

Давайте розглянемо, що такого особливого в мові JavaScript, чого ми можемо досягти за її допомогою та які ще технології пов'язані з нею.

Що таке JavaScript?

JavaScript було створено для того, щоб "оживити вебсторінки".

Програми цією мовою називаються *скриптами*. Їх можна писати прямо в коді HTML-сторінок і вони автоматично виконуватимуться при їх завантаженні.

Скрипти записуються та виконуються як простий текст. Для запуску їм не потрібна спеціальна підготовка чи компілятор.

У цьому плані JavaScript дуже відрізняється від іншої мови програмування — Java .

> **Чому цю мову називають JavaScript?**
>
> Коли мову JavaScript було створено, спочатку вона мала іншу назву: "LiveScript". Тоді була дуже популярна мова програмування Java, тому було вирішено, що позиціонування нової мови як "молодшої сестри" Java допоможе в її популяризації.
>
> Згодом JavaScript стала повністю незалежною мовою програмування зі своєю специфікацією ECMAScript і зараз не має нічого спільного з Java.

Сьогодні JavaScript може виконуватися не тільки в браузері, але й на сервері або на будь-якому пристрої, який має спеціальну програму — рушій JavaScript .

Браузер має вбудований рушій, який інколи називають "віртуальною машиною JavaScript".

Різні рушії мають різні "кодові назви". Наприклад:

- V8 – в Chrome, Opera та Edge.
- SpiderMonkey – в Firefox.
- …Є також інші кодові назви як "Chakra" для IE, "JavaScriptCore", "Nitro" і "SquirrelFish" для Safari, та інші.

Терміни вище добре було б запам'ятати, оскільки вони використовуються в статтях розробників на просторах інтернету. Ми також будемо їх використовувати. Наприклад, якщо "можливість X підтримується в V8", то, імовірно, вона працюватиме в Chrome, Opera та Edge.

Як рушії працюють?

Рушії складні. Але принцип роботи простий.

1. Рушій (вбудований, якщо це браузер) читає (“розбирає”) скрипт.
2. Потім він перетворює (“компілює”) скрипт у машинний код.
3. Після чого цей машинний код виконується, причому досить швидко.

Рушій застосовує оптимізації на кожному етапі процесу. Він навіть слідкує за скомпільованим скриптом під час його виконання, аналізуючи дані, що проходять через нього, і оптимізує машинний код, зважаючи на ці знання.

Що може вбудований у браузер JavaScript?

Сучасний JavaScript – це “безпечна” мова програмування. Вона не надає низькорівневого доступу до пам’яті чи процесора, оскільки була створена для браузерів, які цього не потребують.

Можливості JavaScript значно залежать від середовища, у якому виконується скрипт. Наприклад, Node.js підтримує функції, які дозволяють JavaScript читати/записувати довільні файли, здійснювати мережеві запити тощо.

Вбудована в браузер JavaScript може робити все, що пов’язано з управлінням вебсторінками, взаємодією з користувачем та вебсервером.

Наприклад, JavaScript може:

- Додавати новий HTML-код на сторінку, змінювати наявний вміст, змінювати стилі.
- Реагувати на дії користувача, опрацьовувати натискання миші, переміщення вказівника, натискання на клавіші клавіатури.
- Надсилати запити мережею до віддалених серверів, викачувати та надсилати файли (так звані технології AJAX і COMET).
- Отримувати і надсилати куки , ставити запитання відвідувачам, показувати повідомлення.
- Запам’ятовувати дані на стороні клієнта (“local storage ”), які будуть доступні в майбутніх сесіях на цьому вебсайті.

Що НЕ може JavaScript?

Можливості JavaScript у браузері обмежені для безпеки користувача. Мета полягає в тому, щоб небезпечні вебсторінки не мали доступу до приватної інформації та не могли пошкодити інформацію на комп’ютері користувача.

Приклади таких обмежень:

- JavaScript на вебсторінці не може читати/записувати довільні файли на жорсткому диску, копіювати їх чи виконувати програми. Скрипт не має прямого доступу до функцій ОС.

  Сучасні браузери дозволяють працювати з файлами, але доступ до них обмежений і надається тільки тоді, коли користувач виконав відповідні дії, наприклад, перетягнув

файл у вікно браузера чи вибрав його через тег `<input>`.

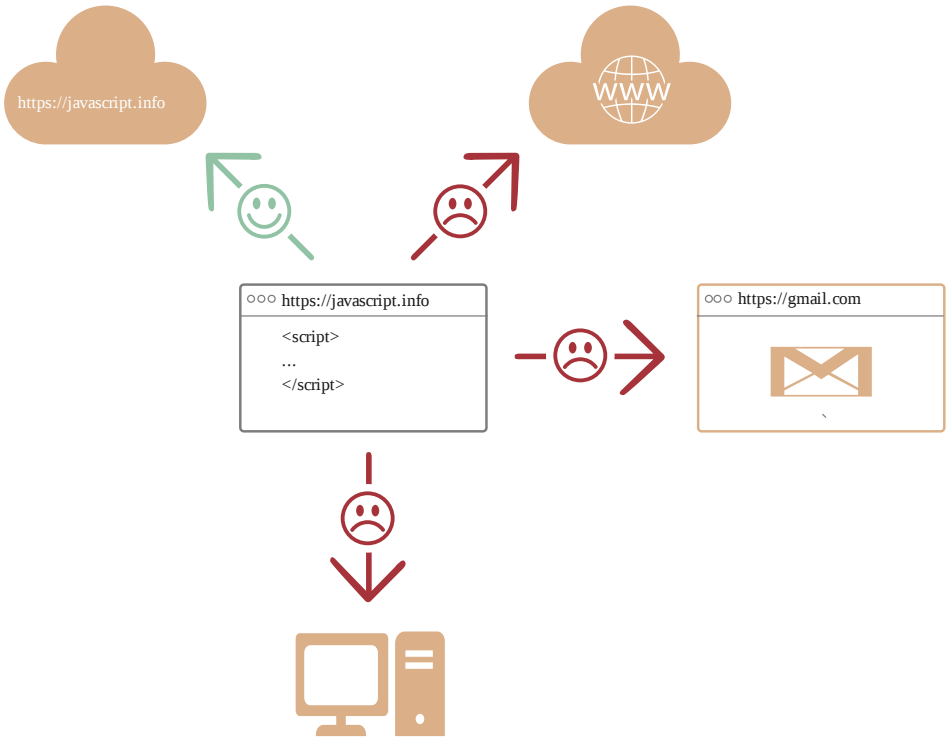
Є шляхи взаємодії з камерою/мікрофоном та іншими пристроями, але для цього потрібен явний дозвіл користувача. Тому сторінка з JavaScript не може нишком увімкнути веб-камеру, спостерігати за оточенням і надсилати інформацію до СБУ.

- Різні вкладки/вікна зазвичай не знають одне про одного. Іноді це можливо, наприклад, коли одне вікно використовує JavaScript, щоб відкрити інше. Але навіть у такому випадку JavaScript з однієї сторінки не має доступу до іншої, якщо вони з різних сайтів (мають різні домени, протоколи чи порти).

  Це називається "Політикою того ж походження (Same Origin Policy)". Щоб обійти це обмеження, *обидві сторінки* мають погодитися на обмін даними та містити спеціальний JavaScript-код, який здійснюватиме це. Ми розглянемо цю тему в посібнику.

  Знову-таки, це обмеження існує задля безпеки користувача. Сторінка за адресою `http://anysite.com`, яку відкрив користувач, не повинна мати доступу до іншої вкладки браузера з URL-адресою `http://gmail.com` і викрадати звідти інформацію.

- JavaScript може легко спілкуватися мережею з сервером, від якого отримана поточна сторінка. Але здатність скрипта отримувати дані з інших сайтів/доменів обмежена. Такі запити можливі, але потребують спеціальної згоди (вираженої в HTTP-заголовках) від віддаленого сервера. Це також зроблено з метою безпеки.



Таких обмежень немає, якщо JavaScript використовується за межами браузера, наприклад, на сервері. Сучасні браузери дозволяють установлювати плаґіни/розширення, які мають розширені можливості, проте вимагають розширених прав.

Що робить мову JavaScript унікальною?

Є принаймні *три* чудові особливості JavaScript:

- Цілковита інтеграція з HTML/CSS.
- Прості речі робляться просто.
- Підтримується всіма сучасними браузерами та є типово увімкненою.

JavaScript – це єдина браузерна технологія, яка суміщає ці три речі.

Це й робить її унікальною. Ось чому JavaScript – найбільш поширений засіб створення браузерних інтерфейсів.

До слова, JavaScript також дозволяє створювати сервери, мобільні застосунки тощо.

Мови “над” JavaScript

Синтаксис JavaScript не задовольняє потреби кожного. Різні люди хочуть різних функцій.

Цього слід очікувати, тому що проєкти та вимоги різні для кожного.

Останнім часом з’явилося безліч нових мов, які *транспілюються* (конвертуються) в JavaScript перед виконанням у браузері.

Сучасні інструменти роблять транспіляцію дуже швидкою та прозорою, дозволяючи розробникам писати код іншою мовою й автоматично конвертувати його “під капотом”.

Приклади таких мов:

- CoffeeScript – це “синтаксичний цукор” для JavaScript. Вона вводить коротший синтаксис, дозволяючи нам писати більш чіткий і точний код. Зазвичай це до вподоби програмістам, які пишуть на Ruby.
- TypeScript зосереджена на додаванні “строгої типізації даних” для спрощення розробки та підтримки складних систем. Розробляється Microsoft.
- Flow також додає типізацію даних, але іншим способом. Розробляється Facebook.
- Dart – це автономна мова, яка має власний рушій, що працює в небраузерних середовищах (як-от мобільні застосунки), але також може транспілюватися в JavaScript. Розробляється Google.
- Brython – це транспілятор коду Python у JavaScript, що дозволяє писати застосунки на чистому Python без використання JavaScript.
- Kotlin – це сучасна, лаконічна та безпечна мова програмування, яку можна компілювати для браузера або NodeJS.

Існують й інші мови. Звісно, навіть якщо ми використовуємо одну з цих транспілюючих мов, ми також повинні знати JavaScript, щоб дійсно розуміти, що робимо.

Підсумки

- Мова JavaScript спочатку була створена лише як мова для браузера, але зараз її також використовують в інших середовищах.
- Сьогодні JavaScript позиціонується як найбільш поширена мова для браузера, яка повністю інтегрована з HTML/CSS.

- Існує багато мов, які “транспілюються” в JavaScript і надають певні функції. Рекомендується переглянути їх, хоча б мигцем, після освоєння JavaScript.

Довідники й специфікації

Цей сайт – *посібник*. Він спрямований на те, щоб допомогти вам поступово вивчити мову. Проте, як тільки ви познайомитеся з основами, вам знадобляться й інші джерела.

Специфікація

Специфікація ECMA-262 містить найглибшу, найдетальнішу й найбільш формалізовану інформацію про JavaScript. Фактично, ця специфікація визначає мову.

Але саме через формалізований стиль її важко зрозуміти з першого разу. Тому, якщо вам потрібне найнадійніше джерело інформації про деталі мови, специфікація – правильне місце. Однак, це джерело не для повсякденного використання.

Щороку випускається нова версія специфікації. Між цими випусками, остання “чернетка” доступна на сайті https://tc39.es/ecma262/ .

Щоб прочитати про найновіші функції, включно з тими, які “майже входять в стандарт” (так звана “стадія 3”), перегляньте пропозиції на https://github.com/tc39/proposals .

Також, якщо ви розробляєте для браузерів, вам буде цікаво прочитати про інші специфікації, які описано в другій частині цього підручника.

Довідники

- **MDN (Mozilla) JavaScript Reference** – це головний довідник з прикладами та іншою інформацією. Він чудово підходить для детального вивчення окремих функцій, методів тощо.

  Його можна знайти за цим посиланням https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference .

Хоча, замість пошуку на сайті, краще використовувати пошукові системи. Просто напишіть “MDN [термін]” в пошуковому запиті. Наприклад, запит https://www.google.com.ua/search?q=MDN+parseInt знайде інформацію про функцію `parseInt`.

Таблиці сумісності

Мова JavaScript активно розвивається – до неї регулярно додаються нові функції.

Щоб дізнатися, чи підтримує браузер або інший рушій певну можливість JavaScript, дивіться на сайтах:

- https://caniuse.com/ – для кожної технології приведено таблицю сумісності з усіма браузерами тобто, щоб побачити, які браузери підтримують сучасні криптографічні функції, слід ввести в пошуку “Cryptography ”.
- https://kangax.github.io/compat-table – таблиця з усіма можливостями мови та рушіями, які підтримують або не підтримують відповідні технології.

Всі ці ресурси корисні в повсякденній розробці, бо містять корисну інформацію про деталі мови, їхню підтримку тощо.

Будь ласка, збережіть собі ці сайти (або цю сторінку) вони вам знадобляться, якщо буде потреба детальніше розібратися в конкретному функціоналі мови.

Редактори коду

Редактор коду – це місце, де програмісти проводять найбільше часу.

Є два основні види редакторів коду: IDE і легкі редактори. Багато людей використовують декілька таких редакторів для різних потреб.

IDE

Термін IDE (Інтегроване середовище розробки) означає потужний редактор з багатьма можливостями, що зазвичай працює з “цілим проєктом”. Як зрозуміло з назви, це не тільки редактор коду, а повноцінне “середовище розробки”.

IDE завантажує проєкт (який може мати багато файлів), дозволяє переключатися між файлами, надає можливість автозаповнення, яке базується на цілому проєкті (не лише на відкритому файлі), інтегрується із системою контролю версій (наприклад, git), надає можливість розгортання вашого проєкту на тестове середовище та багато інших функцій “на рівні проєкту”.

Якщо Ви досі не вибрали IDE, розгляньте наступні варіанти:

- Visual Studio Code (багатоплатформний, безкоштовний).
- WebStorm (багатоплатформний, платний).

Для Windows, також може бути “Visual Studio”, не плутайте з “Visual Studio Code”. “Visual Studio” – потужний платний редактор, який працює лише на Windows, добре підходить для програмування на платформі .NET. Також хороший для програмування на JavaScript. Також існує його безкоштовна версія: Visual Studio Community .

Багато IDE платні, проте мають пробний період. Їхня вартість зазвичай незначна в порівнянні із зарплатою кваліфікованого розробника. Правильний вибір редактора дозволить зберегти найцінніший ресурс – ваш час. Тому просто виберіть найкращий варіант, який задовольнятиме усім вашим потребам.

Легкі редактори

“Легкі редактори” не такі потужні, як IDE, проте вони прості, доступні і швидко запускаються.

Їх зазвичай використовують, щоб швидко відкрити і відредагувати один або декілька файлів.

Головна їхня відмінність від IDE в тому, що IDE працює на рівні проєкту, тому він завантажує набагато більше даних під час запуску, і якщо потрібно, аналізує його структуру. Легкий редактор набагато швидший, якщо нам необхідно відредагували лише один файл.

На практиці, легкі редактори можуть мати багато плаґінів, включаючи аналізатори синтаксису на рівні проєкту, автозаповнення і т. д. Через те, що це значно розширює їх можливості, немає чіткої межі між легкими редакторами та IDE.

Ось ці варіанти заслуговують вашої уваги:

- Sublime Text (багатоплатформний, безкоштовний на час випробувального терміну).
- Notepad++ (Windows, безкоштовний).
- Vim та Emacs також хороші, якщо знати, як ними користуватися.

Не будемо сперечатися

Я, та мої хороші друзі-розробники, вже давно користуємося цими редакторами, і вони цілком задовольняють усім нашим потребам.

У нашому великому світі є й інші редактори. Будь ласка, приділіть трохи часу на ознайомлення з різними редакторами, і виберіть той, який вам найбільш до вподоби.

Вибір редактора, як і будь-якого іншого інструменту, індивідуальний, і залежить від ваших проєктів, звичок і персональних вподобань.

Особиста думка автора:

- Я б використовував Visual Studio Code , якщо розробляти доводиться переважно фронтенд.
- В іншому випадку, якщо це здебільшого інша мова/платформа та лише частково фронтенд, тоді розгляньте інші редактори, такі як XCode (Mac), Visual Studio (Windows) або сімейство Jetbrains (Webstorm, PHPStorm, RubyMine тощо, залежно від мови).

Інструменти розробника

Будь-який код схильний до помилок. Швидше за все, ви будете робити помилки… Хоча, про що я говорю? Ви *точно* будете робити помилки, принаймні, якщо ви людина, а не робот .

Зазвичай, користувачі не бачать помилок у браузері. Тому, якщо в скрипті трапиться щось хибне, ми не побачимо помилки і не зможемо її виправити.

Щоб побачити помилки і отримати додаткову інформацію про виконання скриптів, було створено і вбудовано в браузери “інструменти розробника”.

Більшість розробників надають перевагу Chrome чи Firefox, тому що ці браузери мають найкращі інструменти розробника. Інші браузери теж мають інструменти розробника, деколи навіть зі спеціальними функціями, проте вони не такі популярні, як Chrome чи Firefox. Тому більшість розробників мають “улюблений” браузер і переключаються на інші, якщо проблема специфічна для браузера.

Інструменти розробника потужні і мають багато функцій. Для початку, ми вивчимо, як їх відкрити, як переглядати помилки і як виконувати команди JavaScript.

Google Chrome

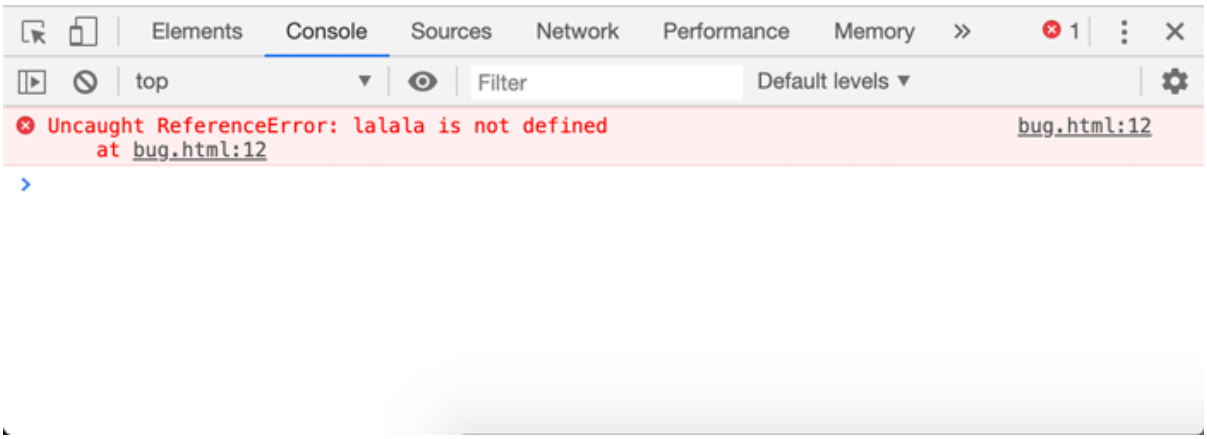
Для прикладів ми будемо використовувати браузер Google Chrome . Інструменти розробника в ньому показуються лише англійською мовою, незалежно від налаштувань браузера.

Відкрийте сторінку bug.html. На ній є помилка в коді JavaScript. Вона прихована для звичайних користувачів, тому потрібно відкрити інструменти розробника, щоб її побачити.

Натисніть клавішу `F12` або, якщо у вас Mac, комбінацію клавіш `Cmd+Opt+J`.

Інструменти розробника типово відкриваються на вкладці “Console” (консоль).

Ось так відображається помилка в консолі:



Точний вигляд інструментів розробника може відрізнятися в залежності від вашої версії Chrome. Вони міняються час від часу, але в основному це вікно повинно бути схожим.

- Тут ми можемо побачити червоне повідомлення про помилку. У нашому випадку, скрипт має невизначену команду “lalala”.
- З правого боку є посилання на джерело `bug.html:12` з номером рядка, де ця помилка виникла. При натисканні на це посилання, вас перенаправить на вкладку “Sources” (файли з кодом сторінки), де відкриється файл і перейде на рядок, в якому трапилася помилка.

Нижче повідомлення про помилку є синій символ `>`. Цей символ позначає “командний рядок”, де ми можемо вводити команди JavaScript. Натисніть `Enter`, щоб їх виконати.

Тепер ми бачимо помилки, цього достатньо, щоб почати. Ми пізніше повернемося до інструментів розробника, щоб розглянути налагодження коду у розділі Налагодження в браузері.

Введення декількох рядків

Зазвичай, коли ми вводимо один рядок коду в консоль і натискаємо `Enter`, він виконується.

Щоб ввести декілька рядків коду, натисніть `Shift+Enter`. Таким чином можна вводити і виконувати довгі фрагменти JavaScript коду.

Firefox, Edge та інші

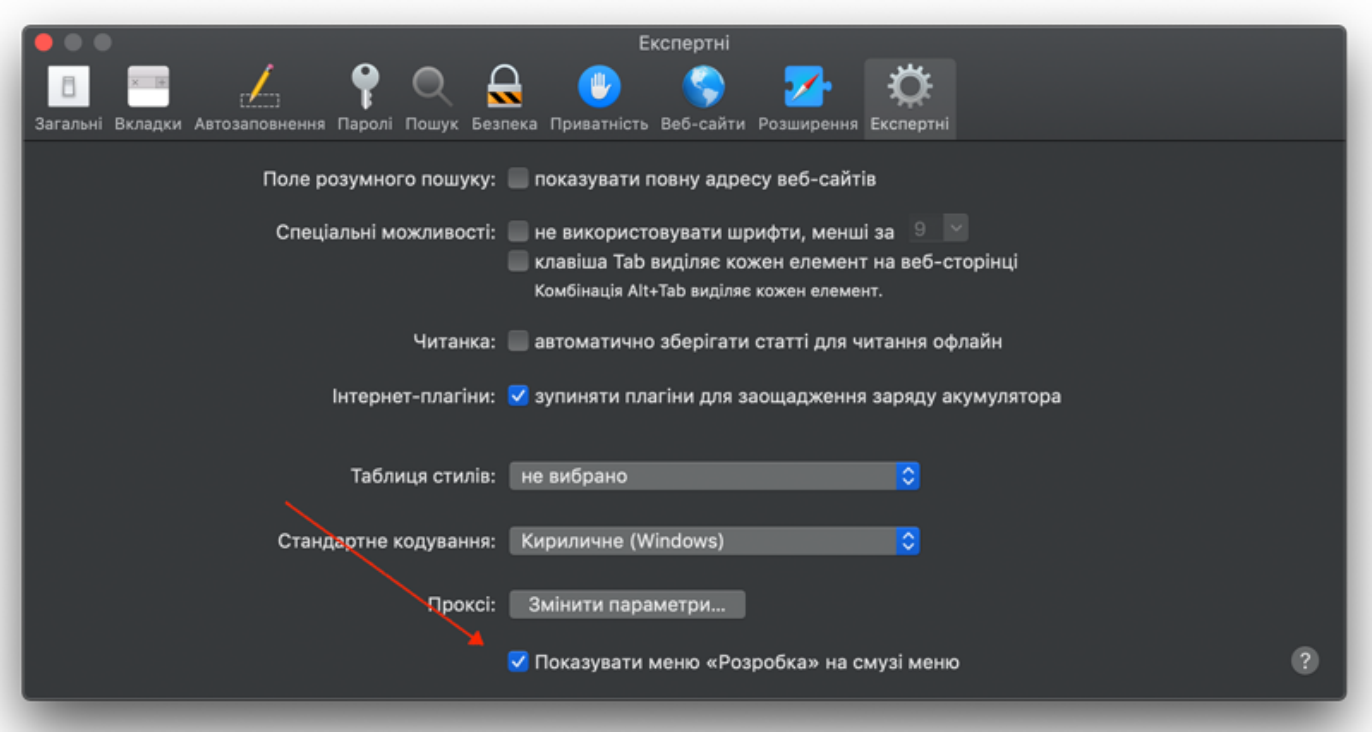
Більшість браузерів використовують клавішу `F12`, щоб відкрити консоль розробника.

Їх вигляд зазвичай схожий. Якщо ви навчитеся використовувати однин з них (можете почати з Chrome), ви зможете легко переключитися на інший браузер.

Safari

Safari (стандартний браузер у macOS, не підтримується Windows/Linux) має свої нюанси. Спочатку нам потрібно увімкнути меню “Розробка”.

Відкрийте Параметри і перейдіть на панель “Експертні”. Знизу буде галочка, яку необхідно вибрати:



Тепер комбінація клавіш `Cmd+Opt+C` може переключати консоль. Також зауважте, що з’явився новий пункт “Розробка” у верхньому меню. Це меню має багато команд та опцій.

Підсумки

- Інструменти розробника дозволяють нам переглядати помилки, виконувати команди, досліджувати змінні та багато іншого.
- Їх можна відкрити клавішою `F12` для більшості браузерів в Windows. В Chrome для Mac потрібно натиснути комбінацію клавіш `Cmd+Opt+J`, в Safari: `Cmd+Opt+C` (але спочатку інструменти потрібно увімкнути).

Тепер у нас є готове середовище. В наступному розділі ми приступимо до самого JavaScript.