Курс AJAX

1. Що таке AJAX
2. Налаштування сервера (приклад)
3. Практика роботи AJAX
   1. Створити об’єкт XMLHttpRequest (приклад)
   2. Розібрати послідовність виконання
   3. GET or POST (приклад як відправляти інформацію)
      1. setRequestHeader
      2. PHP
   4. Access Across Domains
   5. Синхронні запити (приклад)
   6. Асинхронні запити (приклад)
4. Властивості об'єкта XMLHttpRequest
   1. readyState
   2. onreadystatechange
   3. status
   4. Обновлення DOM (приклад 3 попередніх параметрів)
      1. responseText (приклад)
      2. responseXML (приклад)
      3. XML DOM Nodes (Text is Always Stored in Text Nodes) (приклад)
      4. Callback (приклад)
5. Інтро в jQuery AJAX, метод load та робота з Database через PHP (приклад довантажити інформацію)
6. jQuery метод GET (приклад з пошуком)
7. jQuery метод POST і AJAX JSON (приклад з реєстрацією)
8. jQuery form using POST (приклад форма)
9. **Що таке AJAX.**

*Для початку я повинен сказати що для повного розуміння цього курсу вам необхідні базові навики в написанні сайтів. А саме такі мови як HTML, JavaScript та PHP. Але я хочу додати, що для розуміння курсу не потрібні дуже глибокі знання у цих дисциплінах. Якщо ви ще тільки знайомитеся з тим як працюють сайти, як відбувається зв'язок з сервером то в вас може викинути питання: «А що ж таке AJAX?». Тому на цьому уроці я коротко розкажу про те як працює браузер і AJAX.*

*AJAX це технологія, за допомогою якої ми можемо змінювати вміст в сторінки нашого веб-сайту беручи дані з сервера, не обновлюючи сторінку щоразу. Це може здаватися чимось не дуже важливим, але коли у вас є веб-сайт, на якому потрібно постійно змінювати дані AJAX це те, що допоможе це зробити.*

*Давайте поговоримо як наш браузер отримує інформацію для відображення сторінки. Коли комп’ютер через браузер запитує сторінку на сайті він виступає у ролі користувача. Запит відправляється на машину, на якій зберігається вся інформація про сайт, ця машина називається сервером. Сервер обробляє запит і відсилає інформацію по сторінку назад до комп’ютера, а комп’ютер, в свою чергу вже виводить цю інформацію.*

*Тепер розглянемо декілька прикладів роботи AJAX. Уявимо, що у вас є веб-сайт, який має щось на зразок системи чату, подібно до того, що є на Facebook, де ви можете писати повідомлення одне одному. Якби не було AJAX вам би доводилося оновлювати сайт кожен раз коли ви хочете побачити нове повідомлення від іншого користувача. Але з використанням AJAX ми можемо обновляти сторінку невеликими частинами, не навантажуючи канал зв’язку з сервером, а також значно скорочуючи час очікування.*

*Тобто, якщо на сайті є велика кількість даних, то для того щоб не затримувати завантаження з першого разу сервер відсилає не всі дані. Під час користування сайтом, користувачу можуть знадобитися додаткові дані. Наприклад ви переглядаєте галерею у соц мережах, ви передивилися всі завантажені фото і хочете побачити більше, тоді сайт довантажує ще фото. Тоді вам потрібно знову зробити запит на сервер і він надішле вам повністю нову сторінку з додатковими даними а це значить що сервер вам посилає дуже багато вже не потрібної вам інформації знову і знову, це збільшує ваш трафік, а також час очікування завантаження сторінки. А для того щоб до завантажити додаткові дані, не завантажуючи цілу сторінку заново, використовується AJAX. Даючи певний запит на сервер через AJAX, сервер відсилає тільки ті шматочки інформації, які потрібні вам саме зараз, не спричиняючи великих затримок. Тобто він дозволяє оновлювати веб-сторінку без перезавантаження самої сторінки.*

*AJAX =****A****synchronous****J****avaScript****A****nd****X****ML.*

*AJAX це не мова програмування, це група технологій яка поєднує об’єкт XMLHttpRequest і JavaScript. JavaScript використовується для того щоб взаємодіяти з користувачем, відсилати запроси, приймати їх і змінювати сторінку відповідно з отриманими даними. JavaScript спілкується з сервером через АПІ використовуючи запити XHR по іншому XMLHttpRequest. Тобто об’єкт XMLHttpRequest використовується для запиту даних з веб-сервера. Ось що означає буква Х.*

*Але деякі люди все-таки путаються думаючи що буква Х означає що AJAX може використовувати лише XML для передачі даних. Насправді ж AJAX може використовувати транспортування даних як звичайного тексту так і тексту JSON.*

*Тепер давайте подивимося як це все працює в комплексі.*

*TODO image*

***How AJAX Works***

**

1. *Викликається функція яка запускає AJAX на веб-сторінці*
2. *Об'єкт XMLHttpRequest створюється за допомогою JavaScript*
3. *Об'єкт XMLHttpRequest надсилає запит на веб-сервер*
4. *Сервер обробляє запит*
5. *Сервер надсилає відповідь у веб-браузер*
6. *JavaScript отримує відповідь*
7. *JavaScript обробляє відповідь і обновлює сторінку в браузері*
8. **Налаштування сервера**

*На цьому уроці ви дізнаєтеся як користуватися локальним сервером на комп’ютері. Як ви вже мабуть знаєте, існує frontend та backend сайту. Для того щоб виконувати операції з сайтом які потребують backend мови програмування нам і потрібен сервер. Для цього ми будемо використовувати програму під назвою XAMPP. Ця програма являється локальним сервером. Вам просто потрібно скачати та встановити цю програму на свій комп’ютер. Вона абсолютно безплатна. Тепер подивимося як запускати сайти на цьому сервері.*

Для цього зайдемо в папку куди ми встановили XAMPP. Потім заходимо в папку htdocs. Саме в цю папку ми повинні будемо переміщати папку з файлами нашого сайту для того щоб можна було запускати його на сервері. Давайте так і зробимо. Візьмемо приклад сайту, який я зробив завчасно, і помістимо його на сервер. В подальших уроках ми будемо використовувати саме цей проект для розбору функцій AJAX.

Але спочатку для прикладу запустимо php файл без серверу. Як бачите браузер не сприймає його як сторінку, тому що це не файл HTML. Давайте тепер запустимо сайт з серверу. Для цього відкриємо XAMPP і запустимо сервер. Впринципі, на даний момент наш сайт не зв’язаний з базою даних, тому для того щоб сервер почав генерувати сторінки ним потрібен тільки Apache. Як працювати з базами даних MySQL через AJAX, ми розберемо в наступних уроках. Тому зараз наберемо в пошуковій стрічці localhost і назву папки в якій знаходиться наш сайт. Як ви бачите, сайт працює. Це значить що сервер генерує сторінки з формату php у формат HTML, який сприймає браузер. Я рекомендую зараз відкрити консоль на вкладці Network і відключити кеш, тоді при відкритій консолі кеш не буде зберігатися і в нас не будуть виникати можливі помилки зв’язані з кешуванням файлів.

1. **Практика роботи AJAX**

*Ми вже розбирали механізм роботи AJAX на минулих уроках і в цьому уроці ми подивимося на те як він працює на практиці і розберемо типи зв’язку з сервером. Для цього зайдемо в проект який ми створили на попередньому уроці. Видалимо все лишнє що тут є і почнемо писати код. Для початку створимо об’єкт XMLHttpRequest, який дозволяє обмінюватися даними з сервером і обновляти частини сторінки без перезавантаження самої сторінки. Зараз я напишу надзвичайно просту програму і потім ми розберемося як вона працює.*

Спочатку ми створюємо об’єкт, потім у функції open задаємо тип зв’язку, формуємо запит, а також задаємо ссилку на файл. Send вже відсилає запит і тільки потім за допомогою параметра onreadystatechange ви можете визначити функцію, яка буде виконуватися, коли ваш комп’ютер отримає відповідь від сервера і тоді JavaScript вже змінює сторінку. В нашому випадку просто виводить текст в консоль. Текст береться з файлу text.txt. Давайте перейдемо до браузера, відкриємо консоль, як бачите програма працює. І зараз ми розберемо методи зв’язку, а отже метод open. Інші методи AJAX будемо вивчати на наступних уроках.

(показувати мене і open(method, URL, async) на екрані)

GET or POST

*І перший параметр що ми розглянемо це метод відправки даних на сервер. Існує два методи відправки даних. А саме метод GET і POST. GET працює швидше ніж POST, і може використовуватися в більшості випадків. Проте, існують ситуації в яких завжди використовуються POST запити, а саме:*

1. *Коли потрібно відправити велику кількість даних на сервер. GET має обмеження на розмір відправлених даних, а POST таких обмежень не має.*
2. *Коли потрібно відправити важливі дані. POST більш надійний і безпечний, ніж GET.*
   1. *Так як через запит GET, деякі передаються через URL. Наприклад вказавши умови пошуку, як це робить всіми улюблений Гугл.*
   2. *Запит POST в свою чергу відправляє дані на сервер в самому запиті, а не через URL.*
3. *І остання з основних ролей для яких слугує POST це оновлення файлів або бази даних на сервері, адже тут дуже важливим являються саме точність і безпека даних.*

*Зараз я трошки зміню текст і поясню як працює передача даних через GET.*

Як бачите, в коді, після URL ми поставили знак питання, чим зробили змінні видимими для коду PHP. Також ми перерахували змінні через знак, який на мові програмування означає та, або також. В коді PHP ми перевіряємо чи об’явлені змінні і тільки потім передаємо відповідь на комп’ютер. Значення змінних з URL в PHP добуваються саме таким чином. Тобто GET нижнє підкреслення, квадратні дужки і змінна в кавичках. Перейдемо в браузер. Нажмемо кнопку. Як бачимо, код виводить відповідь у консоль.

Тепер перейдемо до метода POST. Зараз я напишу код і ми його розберемо. В методі POST все вже трохи інакше. В URL ми нічого не пишемо, але пишемо всі параметри в методі сенд. Також перераховуючи їх через знак and. Зверніть увагу що для роботи метода POST необхідно між методами open та send вказати метод setRequestHeader який задає значення заголовка запиту HTTP. В цей метод ми передаємо два параметра. Ім’я заголовка для якого повинно бути вказаного значення. І другий параметр це саме значення. Для метода POST ми ставимо ім’я хедера Content-Type і значення хедера application/x-www-form-urlencoded. Про інші хедери можете почитати в інтернеті. В коді PHP все аналогічно крім методу отримання змінних. Тут замість GET просто стоїть POST. Перейдемо в браузер. Протестуємо код. Як бачимо, код виводить привітання у консоль.

*Access Across Domains*

*Наступний параметр для розгляду це URL, тобто місце розташування файлу. З міркувань безпеки сучасні веб-браузери не дозволяють робити запити на зовнішні домени. Це означає що якщо ви захочете зробити запит на файл, він повинен знаходитися на вашому домені.*

*Синхронні асинхронні запити*

*Перейдемо до третього параметра. Він відповідає за принцип за яким буде виконуватися запит, а саме він буде виконуватися синхронно чи асинхронно.*

*Sync*

*Щоб виконати синхронний запит, на місці третього параметра треба ставити false. Це означає що ви відмінили асинхронну відправку даних. Якщо ви це зробили, то JavaScript буде очікувати відповіді з сервера замість того щоб продовжувати виконувати код далі. Оскільки код очікує відповіді сервера то в функції onreadystatechange немає необхідності. Синхронні запити в процесі виключення з веб-стандартів, але цей процес може зайняти багато років. Тому я заохочую вас використовувати тільки асинхронний метод передачі даних.*

Зараз я вам напишу дуже простий код. Як ви бачите тут стоїть false, а параметр onreadystatechange відсутній. В цьому файлі кнопка synchronous визиває функцію в якому цикл створює 10 синхронних AJAX запитів. Тепер давайте перейдемо в браузер, відкриємо панель розробника на сторінці Console, нажмемо кнопку. Як бачимо код вивів нам 10 значень файла sync.txt. Тепер перейдемо до вкладки Network, приблизимо. Як бачимо всі запити виконуються один за одним, тобто JavaScript не продовжує виконуватися, поки не отримає відповідь від сервера.

*Async*

*Запити сервера повинні надсилатися асинхронно. Щоб виконати асинхронний запит, на місці третього параметра треба ставити true. Роблячи асинхронний запит, JavaScript в процесі очікування відповіді з сервера він може виконувати інші скрипти при очікуванні відповіді сервера, в цьому й є перевага асинхронних запитів.*

Зараз я вам напишу приклад асинхронного запиту на сервер. В цьому коді на місці 3 параметра в методі open поставимо true. І для запобіганню збоїв з виведенням інформації поставимо затримку в 5 Мілі секунд перед виконанням коду в циклі. Це ми зробимо за допомогою метода setTimeout. В цьому коді вже також з’явився параметр onreadystatechange, який необхідний для виведення даних при асинхронному запиті. Тепер знову перейдемо в браузер, відкриємо панель розробника на сторінці Console, нажмемо кнопку. Як бачимо код вивів нам 10 значень файла async.txt. Тепер перейдемо до вкладки Network, приблизимо. Як бачимо всі запити виконуються не чекаючи поки прийде відповідь на попередній запит, тобто JavaScript продовжує виконуватися, очікуючи відповідь з сервера.

1. **Властивості об'єкта XMLHttpRequest**

*На сьогоднішньому кроці ми подивимося як змінювати сторінку через AJAX, а також розберемо характеристики об'єкта XMLHttpRequest.*

*Показати на екрані табличку з статусами*

*Почнемо ми з характеристики readyState. AJAX запит відбувається в 5 етапів і властивість readyState відповідає за етап на якому знаходиться запит. Відрахування етапів починається з 0 етапу, а саме запит не створено, потім 1 - встановлено з'єднання з сервером, 2 - запит отримано сервером, 3 - обробка запиту, і останній 4 – комп’ютер отримав відповідь на запит.*

*Наступна характеристика це onreadystatechange. Ця характеристика визначає функцію, яка буде викликана при зміні параметра readyState. Саме в функції яку визначає ця характеристика ми будемо писати код для зміни сторінки.*

*Наступна характеристика це status об’єкту XMLHttpRequest. Статусів цього об’єкту надзвичайно багато і всі з них ми розглядати не будемо, але основні це 200 – запит відбувся без помилок, 403 – немає доступу до файлу, 404 – сторінку не знайдено.*

responseText

І так, почнемо писати код. В цьому прикладі розберемо роботу з відповіддю в форматі тексту, отже і як працювати з responseText. Повидаляемо лишні файли та код, створимо файл text.txt. Тепер створимо кнопку для виклика нашої кнопки, а також дів з id myDiv, який ми вказали в скрипті. Створимо функцію, назвемо її textUpdate. В ній визначимо об’єкт XMLHttpRequest, визначимо функцію через onreadystatechange. І тепер тут зупинимося. Як ми тільки що вивчили, що коли readyState становить 4 і status становить 200, то запит готовий. Тому вкажемо це в умові if. В if напишемо document.getElementById("myDiv").innerHTML = this.responseText. this в даному випадку означає об’єкт XMLHttpRequest. В методі open поставимо GET, "text.txt" і третім параметром true. Файл text.txt я вже підготував. Допишемо метод send. Перейдемо до браузера і подивимося що в нас вийшло. Як бачите, JavaScript виводить на екран вміст файлу text.txt.

Тепер знову перейдемо до коду. Наша функція виводить на екран текст тільки у випадку коли все пройшло успішно. Давайте тепер додамо код, який буде оповіщати, коли щось пішло не по плану. Додамо else if наприклад коли файл не знайдено, тобто статус = 404, і всередині нипишемо alert("something went wrong");. Тепер змінимо URL на неправильний і подивимося що буде. Перезавантажимо сторінку в браузері. Відкриємо консоль, нажмемо кнопку. Як бачите з’явилося вікно з нашим повідомленням про те що щось пішло не па плану. В консолі нам показує код 404, що означає, що сторінка не знайдена, бо ми вказали неправильний URL. Взагалі, можна вручну вказувати що буде робити сайт при різних статусах, але ми цим займатися не будемо. Давайте змінемо URL на правильний.

responseXML

Тепер розглянемо як користуватися responseXML. Скопіюємо попередню функцію. Пере назвемо її на xmlUpdate. Створимо нову кнопку для виклику цієї функції. Назвемо Update by readyXML. У функції змінимо URL на food.xml. файл в форматі XML я завчасно скачав з інтернета, просто щоб продемонструвати роботу з ним. Об’явимо змінну, яка буде вміщати відповідь сервера у форматі XML. Зверніть увагу, що тут ми пишемо responseXML, а не responseText. Щоб використовувати responseXML, потрібно щоб сервер відсилав відповідь в структурі XML файлу. Об’явимо змінну для тексту. Для роботи з XML можна використовувати ті самі методи що й для роботи з тегами в HTML. Тому створюємо змінну х дорівнює xmlDoc.getElementsByTagName("name"). Зверніть увагу, що саме Elements, тобто множина.

Ми взяли name, тепер давайте подивимося на структуру файла XML. Кожен тег в XML називається Node. Тобто вимальовується ієрархія з ноудів. У кожного ноуда є діти, тобто вищі по ієрархії ноди. Ми об’явили змінну х, яка являється масивом даних. Тепер зробимо цикл по довжині цього масиву. В змінну xmlText, яка буду зберігати наш кінцевий текст кожен раз будемо додавати наступний елемент масиву х. Який в свою чергу містить один ноуд тексту. Це ми виразимо через метод childNodes з індексом 0. Але якщо ми залишимо так, то сайт буде виводити не саме значення назви, а нод назви, тобто текст в кавичках, тому ми ще допишемо метод nodeValue. Ну і додамо відступ. І в кінці виведемо отриманий текст в дів з id myDiv. Подивимося що виведеться на сайті. Як бачимо, в нас вивівся список назв блюд.

XML DOM Nodes

Повернемося до коду, трохи його змінимо. Name замінимо на food, перед childNodes з індексом 0 додамо ще одні childNodes з індексом 1. Подивимося що вивело в браузері. Вивід не змінився. Ми перейшли на нижчу степінь ієрархії. І вивели значення першого нода, але тоді чому ми поставили індекс 1. Тепер давайте знову перейдемо до XML файлу. Те що я тільки що описав є поширеною помилкою обробки DOM XML-ля, що є очікування того, що нод елемент містить текст, але це не так. Проте текст нода зберігається в окремому текстовому ноді. Елемент вузла <name> містить текстовий нод з значенням назви, але нод з значенням назви не є значенням елемента <name>. Я знаю що спочатку це може бути складним для розуміння, але саме тому ми вказали індекс 1, бо індекс 0 вказував би на нод <name>, значення якого не задане.

Callback

Тепер давайте знову скопіюємо якусь функцію. Змінимо назву на наприклад domUpdate. Задамо вхідні параметри URL та func. Видалимо весь текст в функції onreadystatechange. поставимо туди func і передамо в неї параметр this. URL замінимо на змінну. Створимо ще дві функції, назвемо їх та передамо параметри xhr. В одну вставимо код для XML, в іншу код для тексту, замінимо this на xhr. Створимо кнопки для виклику функції domUpdate з різними вхідними параметрами. В одну помістимо ссилку на текстовий документ і функцію для виводу тексту. В іншу ссилку на XML документ і функцію для виводу XML. Перейдемо до браузеру, обновимо сторінку, як бачимо, вони функціонують так само як і попередні кнопки.

*Отже ми зробили те саме, але іншим способом. Виявляється що в JavaScript функцію можна передавати як параметр іншої функції. Такі функції називаються Callback функціями, або словами функція зворотного виклику. Те що ми зробили й називається використання Callback функції. Якщо на веб-сайті є кілька завдань в яких використовується AJAX, то краще створити одну функцію для створення об'єкта XMLHttpRequest та одну функцію зворотного виклику для кожного завдання зв’язаного з AJAX. Тобто не потрібно кожного разу створювати об'єкт XMLHttpRequest. Виклик функції повинен містити URL-адресу та яку функцію викликати для обробки відповіді з сервера.*

1. **jQuery**

*$(selector).load(URL,data,function(response,status,xhr))*

*На цьому уроці ми розглянемо jQuery AJAX, метод load та роботу з базою через PHP. Оскільки запис крос браузерного універсального коду AJAX може бути достатньо складним, а особливо новачку, оскільки різні веб-переглядачі мають різні синтаксичні можливості для реалізації AJAX. Це означає, що вам доведеться написати додатковий код для тестування для різних веб-переглядачів. А jQuery – це фреймворк який виправляє проблеми сумісності які є у JavaScript в застарілих браузерів також в цій бібліотеці є багато додаткових інструментів для роботи з DOM, а одна з моїх улюблених фич це легкість роботи з AJAX.*

*І перший метод який ми сьогодні розглянемо це метод лоад. Цей метод оновлює HTML селектора, до якого ми його приміняємо. В цей метод ми можемо передавати три параметра, перший це URL і він являється обов’язковим. Інші два обов’язковими не являються. Другий це дані які ми хочемо відправити на сервер. І тут треба відмітити, що через метод лоад можна відправляти дані і через URL, тобто методом GET, або через другий параметр методом POST. І третій параметр визначає функцію для запуску після завершення методу запиту на сервер. Яка також приймає 3 параметра, а саме response – що містить дані результату запиту, status – що містить статус запиту наприклад "success", "error", "timeout", чи інші ("notmodified", "parsererror"), і останній параметр xhr – що містить об'єкт XMLHttpRequest.*

HTML

Сьогодні ми створимо примітивну програму для дозавантиження коментарів з бази. Ми також будемо використовувати jQuery, я вже підключив цей фреймворк, а ви вже самі можете самостійно або скачати його, або як я підключити зовнішню бібліотеку. Почнемо написання коду з HTML. Створимо дів, з айді text\_area. В ньому ще один блок з класом text\_block. І в ньому параметри комента. Напишемо css щоб наші коменти хоч адекватно виглядали. Зробимо скролл, приберемо на тегах тексту великі марджіни і додамо їх до тегів блока.

MySQL

Для цього урока нам потрібно буде включити нашу базу даних на локальному сервері. Тепер перейдемо до бази. Туди можна перейти по кліку на адмін. Створимо нову базу, назвемо AJAX\_lessons, створимо табличку додамо id з auto increment і PRIMARY KEY. Також стовпчики name та description коментаря. Додамо дані комента, ми зробимо їх такими самими як і в нашому HTML, просто якщо робити різні то довелося б додавати код в наш індексовий файл для виведення першого комента. Зробимо чимало коментів для більшої наглядності.

jQuery

Почнемо написання скрипта. Додамо основний тег jQuery. Створимо змінну для підрахунку кількості коментів. Почнемо з одиниці, бо ми один комент вже написали в HTML. Тепер робимо селектор на нашу кнопку, щоб по її кліку спрацьовувала функція. Робимо інкремент на змінну commentcount. Тепер робимо селектор на text\_area. Упс, у нашої кнопки id буде не text\_area, а text\_load. Тепер дописуємо метод load. Першим параметром пишемо ссилку. Наступним параметром передаємо змінну commentcount і давайте іще якусь вигадану змінну. Наприклад somerequest. Як бачите після попередньої змінної потрібно обов’язково поставили кому, але коми не повинно бути після останньої змінної. І третім параметром передаємо функцію з параметрами result, status і xhr. Ну і давайте тут напишемо що якщо статус дорівнює success, то ми виведемо в консоль status і xhr.status. Там де ми написали status, повинно вивести success, а там де xhr.status повинно вивести 200, що теж означає success.

PHP

Тепер перейдемо до php. Спочатку перевіримо чи передалися нам параметри. Це ми зробимо через isset, та перерахуємо параметри через кому. Тепер нам потрібно зробити підключення до сервера. Створюємо змінні servername, username, password і database. В servername пишемо localhost, в username іде root, це дефолтний користувач на сервері xampp. В пароль нічого не вносимо. І в нашому випадку database називається ajax\_lessons. Тепер створюємо об’єкт підключення до бази вносимо параметри саме в такому порядку. потім створюємо змінну, яка буде утримувати значення кількості коментарів, які нам потрібно витягти з бази. Потім створюємо запит SQL. Вибираємо стовпчики name та description з нашої таблиці і вказуємо ліміт на кількість вибраних коментарів ну і вставляємо туди змінну commentcount. Створюємо змінну для утримання результату запиту на базу через функцію mysqli\_query першим параметром якої іде об’єкт бази і другим запит SQL. Потім створюємо цикл, якщо грубо сказати то поки є рядки в масиві даних цей цикл їх виводить. Ми це зробили через функцію mysqli\_fetch\_array, першим параметром якого є результат запиту на базу даних, а другим вид масиву який він буде створювати, ми вибрали асоціативний масив, але можна також було вибрати нумерований масив. В циклі через echo виводимо текстовий блок який ми створили в HTML, але вже з нашими параметрами.

Test

Я забув додати кнопку для запуску функції. Тому створимо кнопку з id text\_load. Тепер перейдемо в браузер відкриємо консоль і подивимося що в нас вийшло. Коментарі виводяться, а в консолі як я і казав виводиться success і 200. Давайте ще понажимаємо кнопку. Ми вивели вже всі коментарі. Як бачите запит відправляється, але вже більше немає коментарів в базі.

1. **jQuery метод GET і AJAX JSON**

*$.GET(URL, function(response,status,xhr));*

*На цьому уроці ми поговоримо про метод GET за допомогою jQuery, попрацюємо з JSON та зробимо простеньку програму пошуку. Перший параметр GET є обов’язковим і являє собою URL. Як ми вже розглядали до цього, дані метода GET передаються через URL після знаку питання розділяючись знаком and і передаючи значення параметра через дорівнює. І другий необов’язковий параметр такий самий як і третій параметр метода load це функція яка виконуються при завершені запиту. Хотілося б повторити, що GET використовується тільки для отримання даних з сервера, і ні в якому разі не для збереження їх. Для збереження даних потрібно використовувати метод POST. Також запит GET може бути збереженим як ссилка, що ми наприклад й робимо коли ми робимо закладки на якісь запити в гугл.*

HTML

І так, почнемо написання нашого пошуку. Ви пам’ятаєте що в нас був файл в форматі XML з даними про їжу, так ось це файл з тими самими даними, тільки в форматі JSON. Тепер почнемо написання коду HTML. Зробимо div з id search\_area. В ньому тег заголовку та тег input. В input поставимо type="search" id="search", в також placeholder="name or description of the dish". Тим хто не знає то placeholder це текст підказка який виводиться в input перед тим як ви почали вводити текст. Створимо також тег з id result\_area для виводу блоків з інформацією. Також створимо приклад блоку куди буде виводитися наша інформація з файлу JSON. Дамо кожному з тегів клас для того щоб можна було налаштувати зовнішній вигляд. Зараз давайте подивимося що в нас вийшло в браузері. Виглядає не дуже. Я зараз швидко напишу css і потім перейдемо до нашого уроку. І так, тепер стало краще. Я не показував як писати css на відео, бо я надіюсь ви вже вмієте це робити, також код буде прикріплений до уроку.

jQuery

Так що почнемо писати скрипт і перше що нам потрібно зробити це перевіряти дані які ми вводимо в input. Для цього підійде метод keyup який запускає функцію кожен раз коли ми вводимо щось в input. Тепер візьмемо значення з input і перевіримо чи все працює. Будемо виводити дані в консоль. І так, все працює, отже все гаразд. Тепер нам потрібно перевести це значення в регулярні вирази, для того щоб запобігти несходження зв’язане з регістром букв. Тобто ставлячи другим параметром «і» ми робимо це значення нечутливим до регістру.

Зараз я почну з використання getJSON, це звичайний запит GET, але просто для передачі даних в форматі JSON, в першу змінну пишу наш файл JSON і дані з цього файлу будуть передані в функцію і оброблені в залежності від введеного тексту і виведені на екран. Тому для початку створюємо змінну для утримання нашого значення для виводу в html. І тепер пишемо функцію пошуку. Для цього добре підійде метод jQuery each, який перебирає всі елементи масива. Пишемо if для перевірки значень в елементах файлу JSON name та description. При чому метод getJSON вже зразу передає нам дані в форматі JSON і нам не потрібно буде переводити звичайний текст в формат JSON. Тому ми беремо змінну val яка зберігає наші дані і пишемо ті значення які нам потрібні. А для перевірки ми використаємо метод search в який вносимо значення яку ввели в input для звіркою з даними JSON. Якщо метод search повертає значення -1, то це значить що жодного збігу не знайдено. Тепер візьмемо html нашої заготовки і замість константних значень вставимо в нього значення з файла JSON. І все це помістимо в змінну htmlVal. І потім виведемо htmlVal в наш div з id result\_area. Ну і видалимо або за коментуємо нашу html заготовку.

Перейдемо до браузера, виправимо отриману помилку. Тепер обновимо сторінку, напишемо щось, як бачите, пошук працює і для назв, і для описів.

1. **jQuery метод POST**

*$.post(URL, data, function(response,status,xhr));*

*На цьому уроці ми подивимося як використовувати метод POST за допомогою jQuery та створимо примітивну систему реєстраціє користувача. Навіть не зовсім реєстрацію, там не буде log in і log out, ми просто будемо зберігати користувача в базі. Просто для створення log in і log out потрібно розуміти що таке Sessions в PHP, а це вже зв’язано більше з розумінням PHP, а не технологією AJAX. Тому, перейдемо вже до самого уроку. В метод POST в бібліотеці jQuery ми передаємо 3 параметра так само як і в метод LOAD. Тобто URL, data і callback функція. Але на відміну від LOAD параметри в методі POST через URL передавати не можна. Впринципі весь основний теоретичний матеріал ми вже вивчили протягом попередніх уроків. Тому давайте одразу перейдемо до написання коду.*

HTML

Зараз по швидкому напишемо HTML структуру для вводу даних. Створимо теги заголовку інпути та кнопку для відправки даних, дамо їм всім id. Та призначимо placeholder інпутам. Також я створив дів для виводу відповіді з сервера. Зараз я напишу швиденько код css для того щоб все це нормально виглядало і перейдемо до JavaScript. І так я написав css, ось так ось це виглядає.

JavaScript

Почнемо писати JavaScript. При нажаті на нашу кнопку буде спрацьовувати функція яка запусте AJAX запит на сервер. Але для цього нам потрібно взяти змінні зі значеннями введених даних. Ми використаємо для цього методи val, які ми вже використовували в попередньому уроці. Тепер перед відправкою запиту перевіримо чи введені якісь дані в наші input. Якщо ні, то виведемо попередження, що треба ввести дані. В нас вже є файл PHP тому запит будемо робити на нього. І так, запит на файл request.php, передаємо дані імені та пароль. І якщо статус дорівнює success, то виводимо дані на екран.

SQL

Перейдемо до бази, в ми працюємо з тією самою базою що й до цього, але в ній потрібно створити нову таблицю. І так почнемо писати SQL код. Взагалі, всі ці операції можна робити за допомогою можливостей phpmyadmin, але мені зручніше писати саме кодом. Тому пишемо CREATE TABLE `ajax\_lesson\_7` створюємо колонки id з типом даних int та з параметрами NOT NULL AUTO\_INCREMENT і PRIMARY KEY. Наступні колонки name та pass без ніяких параметрів та з типом даних varchar(), зробимо розміри 255. З базами закінчили.

PHP

Тепер перейдемо вже до оформлення запиту в PHP. Підключення до нашої бази ми вже робили, тому якщо ви нічого не міняли, можете залишити все як є. перевіряємо чи передалися наші дані на сервер. Створюємо змінну sql, та запит в ній. В цьому запиті ми будемо вводити наші значення в рядки нашої таблиці ajax\_lesson\_7. Використаємо метод mysqli\_query для проведення запиту на базу. Передамо туди параметри. Перший – об’єкт підключення до бази, другим нашу змінну sql. А знаєте, краще напишемо перевірку на помилки. Якщо наш запит на базу пройшов успішно, а отже дорівнює TRUE, то що New user created successfully, тобто новий користувач успішно створений. Якщо ні, то виведемо Error, нашу змінну з кодом sql та методом mysqli\_error, з переданим йому об’єктом conn. Цей метод виводить можливі помилки в запиті на базу даних. Ми ще забули призначити нові змінні для переданих від користувача. Тепер все готово.

TEST

Давайте перевіримо що в нас вийшло, перейдемо на сайт. Створимо якогось користувача. Так, перейдемо до бази, обновимо, і ми бачимо що користувача створено, прекрасно.

PHP

Тепер зробимо так щоб нам виводило дані користувача, з бази на екран. Назначимо на sql новий запит через SELECT вибираємо всі дані користувача там де його ім’я дорівнює імені користувача, якого ми щойно створили. Виконаємо запит, прирівняємо до result. І через цикл while виведемо всі дані. Ми вже робили таке на попередніх уроках, тому не буду зациклюватися на цьому. Але повторюсь що пишемо назву колонки в квадратних скобках.

TEST

Перевіримо що в нас вийшло. Так, зареєструємо користувача, вуаля нам вивело дані зареєстрованого користувача, разом з кого id. А давайте зробимо помилку в запиті там де ми його перевіряли. Перезавантажимо і спробуємо зареєструвати знову. Як бачите, зараз нам виводить помилку і звіт про неї. Давайте виправимо помилку, і зробимо все як було. Тепер давайте спробуємо зареєструвати декілька однакових користувачів, як бачите, нам це вдалося. Тому давайте це виправимо.

JavaScript 2

Для цього в JavaScript створимо глобальну змінну, яка буде утримувати значення того чи існує введене ім’я чи ні. Призначмо змінній false. Тепер використаємо метод, який вже використовувався в попередньому уроці, коли ми робили пошук. Це метод keyup. Отже кожен раз коли текст в input імені змінюється викликається функція, яка робить POST запит на базу і перевіряє чи існують такі імена. Можна просто скопіювати і змінити вхідні дані з попереднього запиту. Ми видалимо дані по паролю і також його змінну. Все інше залишимо як є.

PHP 2

Перейдемо в PHP. Зробимо elseif при існуванні тільки $\_POST['name']. Прирівняємо змінну імені. Просто скопіюємо попередній запит. І перевіримо чи є якісь результати за допомогою метода mysqli\_num\_rows. Тобто якщо немає переданих рядків, то ім’я вільне. Тому якщо ім’я вільне виведемо This name is free, в іншому випадку виведемо This name is already in use. Тепер будемо на основі отриманої від сервера відповіді динамічно міняти змінну nameFree. Якщо відповідь рівна This name is free, то nameFree = true. А якщо This name is already in use, то nameFree = false.

Давайте перевіримо за допомогою console.log чи працює наш код. Ага, ми ще забули видалити перевірку на наявність пароля. Тепер перейдемо до браузеру, відкриємо консоль. Давайте когось зареєструємо. Тепер введемо його ім’я, показує false, значить ім’я заняте, введемо якесь інше, показує що ім’я вільне. Прекрасно, код працює.

Тому, додамо перевірку цього параметра в запит на збереження даних, там де ми перевіряємо заповненість полів вводу. Давайте знову перейдемо до браузера, введемо зареєстрованого користувача, нажмемо зареєструватися. Не реєструє, введемо якогось незареєстрованого. Вуаля, зареєструвало, отже все прекрасно працює.

1. **jQuery form using POST**

*Це завершальний урок з серії про технологію AJAX. І на цьому уроці я покажу приклад форми зворотного зв’язку з використанням AJAX, метода POST. Також дані ми будемо передавати у форматі JSON. Для цього уроку я вже написав css, щоб зекономити ваш час, можете його продивитися зараз, а також скачати з архіву під відео. Тому перейдемо до написання.*

HTML

І почнемо ми як завжди з HTML. Ми створимо форму, вкажемо action="request.php", method="POST". В звичайних формах можна використовувати передачу даних з форми до PHP через action, але тоді сервер не зможе передати результати на цю саму сторінку, тому доведеться перезавантажувати сайт. Я покажу як цього уникнути, навіть якщо ви вказали action. Але це вже коли ми почнемо писати JavaScript. Тепер додамо два input, перший для email другий для імені. Але в обох вкажемо тип текст, бо якщо для input ми вкажемо тип email, дефолтна заборона на введення неправильного типу даних буде заважати нам виконати власну, тобто кастомну. Якщо хочете, можете з цим проекспериментувати. Тепер зробимо вибір статі. Зробимо це через select разом з option. Зробимо опцію зі значенням чоловік та жінка. Та надамо value ці значення, але англійською мовою. І надамо опції зі значенням чоловік вибір за змовчуванням за допомогою параметра selected. Створимо textarea. Також створимо кнопку для відправки повідомлення з типом submit. Тепер нам потрібно зробити div для виводу повідомлень з сервера. В цьому div зробимо два макети повідомлень через span, один для успішного завершення, другий для помилки. І для завершення роботи з HTML, призначимо кожному з наших тегів свій унікальний id. І я вже завчасно прикріпив файл з стилями.

TEST

Тепер перевіримо чи все працює в браузері. Перейдемо на нашу сторінку та перезавантажимо. Як бачимо, всі стилі працюють, але наша форма нічого поки-що не робить, а коли ми нажимаємо на кнопку, то нас просто перекидає на файл request.php.

JavaScript

Перейдемо до JavaScript. Створимо селектор на нашу форму, з id form-box. І коли ми нажимаємо на кнопку з типом submit, яка знаходиться в цій формі, викликаємо функцію, в яку передаємо event. І для об’єкта event примінимо метод preventDefault. Event – це дефолтна функція яка би запустилася, який ми її не відмінили методом preventDefault. Тепер назначимо змінну для кожного з полів нашої форми, а також вибірки і кнопки. Тепер створимо запит POST, куди передамо всі тільки но створені змінні. У callback функцію передамо дані отримані з сервера. І так як ми знаємо що ці дані будуть у форматі JSON одразу їх переведемо у зручну для користування форму через метод JSON.parse, в який передаємо дані в форматі JSON. Передаючи вже зручні для користування дані в об’єкт response, ми можемо викликати **параметри цього об’єкта** через крапку. В нашому випадку, дані які ми будемо передавати на вивід комп’ютером будуть міститися в параметрі html. Тому виведемо їх в наш div для виводу повідомлень.

PHP

Тепер почнемо програмувати PHP файл. Одразу створимо асоціативний масив для подальшого його перетворення в код JSON. Параметр html буде утримувати дані для виводу. errorFill дані про незаповненість полів. І errorEmail про правильність написання email. Зробимо перевірку на те чи передалися дані. Зробимо це через функцію isset і переданий в неї параметр submit, тобто нашу кнопку. Пере назначимо всі значення з POST на наші змінні. Тепер зробимо перевірку, якщо всі поля пусті. Ми зробимо це через стандартну функцію PHP empty. В яку передамо змінну. Повторимо це для всіх змінних окрім gender, так як вона і так по дефолту має значення. Якщо всі параметри пусті, то призначимо параметру html нашого масиву response текст з макетом помилки, який ми зробили раніше, та змінимо вихідні дані на «Заповніть всі поля». Також змінимо параметр errorFill на true, тобто не всі поля заповнені.

Перейдемо до наступної умови. В ній ми перевіримо чи правильно користувач ввів свій email. І зробимо ми через функцію filter\_var, де перший параметр це email, а наступний це параметр по якому функція буде перевіряти передану їй змінну. В нашому випадку це FILTER\_VALIDATE\_EMAIL, бо ми перевіряємо правильність написання email. І нам потрібно щоб умова виконувалася коли email введений неправильно, тому ставимо знак оклику попереду. В середині умови. Параметру html нашого масиву response призначимо інший відклик, а саме «Введіть правильний email». Також параметр errorEmail пере назначимо на true, тобто проблема з email.

Тепер створимо третю умову, коли все добре. Параметру html передамо повідомлення що все добре і «Повідомлення відправлено», також змінимо там id з form-error на form-success. Тепер перетворимо асоціативний масив в формат JSON за допомогою функції json\_encode, куди передамо наш масив response. І виведемо через echo. Якщо ж проблема виникла на іншому рівні, а саме не передані дані про submit, то виведемо що просто «Виникла помилка».

JavaScript 2

Тепер знову перейдемо до JavaScript і змінимо нашу сторінку у відношенні до отриманого результату. І перша умова, це коли в нас не заповнені поля, тобто response.errorFill == 'true'. Тому для всіх трьох полів вводу ми додаємо клас 'error', цей клас обводить поля червоним кольором. Наш текст html ми вже вивели одразу після розкодування. Інша умова коли неправильно записаний email, тобто response.errorEmail == 'true'. В такому випадку, ми прибираємо клас 'error' з усіх полів вводу і додаємо його для поля з емейлом. І для третього випадку, коли все добре, тобто і response.errorEmail, і response.errorFill дорівнюють 'false'. В такому випадку ви переводимо всі поля до їх дефолтного значення. Для полів це відсутність тексту, а для селектора, це male. І також для всіх полів вводу прибираємо клас 'error'.

TEST 2

Тепер перейдемо до браузера і подивимося результат. Коли ми нікого не ввели і нажали кнопку. Нам виділило всі поля червоним і вивело повідомлення про те що потрібно заповнити всі поля. Якщо ми заповнимо всі поля, але введемо неправильний формат email, то виділить тільки email і скаже що потрібно ввести правильний формат email. Якщо введемо все правильно і виберемо жіночу стать, то нам скаже що повідомлення відправлене, потім зітре всі значення і змінить жіночу стать на чоловічу. На цьому наші уроки закінчуються бажаю вам успіхів в подальшому вивченні wed-development.