**1)** jQuery - швидка, компактна, багатофункціональна і найпопулярніша бібліотека JavaScript.

Її основна ідея – пишучи менше, робити більше

Вона робить зручними такі речі, як обхід HTML-документа та маніпуляція його елементами, обробка подій, анімація та Ajax-запити. Все це здійснюється за допомогою зручного API, який також працює на різних браузерах.

jQuery настільки популярна, що її часто розглядають як невід'ємна складова мови JavaScript, хоча насправді вона такою не є і має бути підключена як будь-яка інша стороння бібліотека.

**2)** Синтаксис jQuery орієнтований на вибір html-елементів та виконання певних дій над ними

Базовий синтаксис виглядає наступним чином: спочатку пишемо знак долара чи назву бібліотеки jQuery повністю, далі у дужках зазначаємо селектор – спеціальний запит, який має повернути html-елементи, що відповідають певній умові, далі через крапку зазначаємо дію – метод бібліотеки, який виконує певну дію.

Саме за таким синтаксисом будується більшість виразів, які використовують jQuery. На слайді я навів кілька прикладів ефектів, які ми можемо робити за допомогою джейквері, трохи детальніше про них на наступному слайді.

**3)** show / hide – показати/приховати елемент, toggle() -перемикання між цими двома методами

fadeIn() використовується для затухання прихованого елемента.

fadeOut()використовується для згасання видимого елемента.

fadeToggle()перемикається між цими двома методами

fadeTo() дозволяє згасати до заданої непрозорості (значення від 0 до 1).

Метод jQuery animate() використовується для створення власних анімацій. Можливості цього методу величезні, ми можемо за його допомогою реалізувати буквально будь-яку анімацію яка нам прийде на думку.

Необхідний параметр params визначає властивості CSS для анімації.

**4)** Далі поговоримо про показ і приховування контенту. jQuery може показувати чи ховати елементи за допомогою методів .show() та hide().

Коли jQuery ховає елемент, бібліотека встановлює властивість display в none. Це означає, що елемент матиме нульову висоту та ширину. Взагалі, я бачив думку на одному з форумів коли про все це читав, що елемент просто стане невидимий та залишає пусту область на сторінці, хоча насправді це не так

На екрані наведено два приклади: перший приховає всі абзаци на сторінці, другий – покаже всі елементи 'div', які мають клас '.hidden'.

**5) Анімований показ та приховування**

**У** jQuery ми можемо показувати та приховувати елементи сторінки, використовуючи ефекти анімації.

Найпростіший варіант використання анімації полягає в тому, що ми передати швидкість, з якою буде виконуватись анімація у якості параметра методу. Для визначення швидкості є три ключові слова: це 'slow', 'normal' та 'fast'.

Якщо існує потреба, то швидкість можна задати в мілісекундах, передавши кількість мілісекунд у якості аргументу методу

**6**) Хотілось би сказати трохи про ефекти появи та зсуву

jQuery використовує комбінацію ефектів появи/зникнення та зсуву, коли показує та приховує елементи. Існує можливість використовувати ці ефекти окремо один від одного.

У наведених прикладах спочатку демонструється анімація зсуву, потім – анімація зникнення. У обох випадках зазначається час в мілісекундах на виконання операції.

**7) Далі поговоримо про зміну відображення елементів на основі стану поточної видимості**

jQuery дозволяє перемикати стан видимості елемента. Тобто, якщо він був прихований, то відбудеться його показ, а якщо був видимий, то, навпаки, відбудеться його приховування. Для цього використовується метод toggle(), який може отримувати ті ж самі аргументи, що ми розглянули раніше.

У наведеному прикладі спочатку відбувається миттєве перемикання видомості всіх абзаців, а потім – повільне перемикання видимості всіх зображень.

Аналогічно існують методи для перемикання за допомогою ефектів зсуву та поступової появи/зникнення.

**ПОКАЗАТИ КОД**

**8)** Дуже важливою частиною jQuery є можливість маніпулювати DOM. DOM це Domain Object Model тобто об’єктна модель документа. Розкажу кілька слів взагалі про це дерево на прикладі роботи джаваскріпта.

Отже, як працює javascript в браузері? Спочатку браузер завантажує сторінку, потім він бере наш html, парсить його та виводить на екран. Після цього він створює DOM дерево

Тобто він переглядає весь наш HTML і створює дерево елементів, з правильною вкладеністю базуючись на HTML, який він відрендерив до цього. Тобто все починається з елемента document, потім у нього вкладається елемент html, куди вкладається head та body. Далі він вкладає кожен елемент один одного в правильній послідовності.

Саме DOM дозволяє нам отримувати доступ до елементів сторінки за допомогою JavaScript.

jQuery постачається з купою пов’язаних з DOM методів, які полегшують доступ до елементів і атрибутів а також маніпулювання ними.

append() – Вставляє вміст у кінець елемента(ів), визначеного селектором.

before() – Вставляє вміст (нові або наявні елементи DOM) перед елементом(ами), який(і) визначено селектором.

after() – Вставляє вміст (нові чи існуючі елементи DOM) після елемента(ів), який(-и) визначено селектором.

prepend() – Вставити вміст на початку елемента(ів), визначеного селектором.

remove() – Видаляє елементи з DOM, указані селектором.

replaceAll() – Замінити цільовий елемент(и) на вказаний.

wrap() - Обгорнути HTML-структуру навколо кожного елемента, визначеного селектором.

**9)** Події джейквері

Що ж таке івенти? Івенти, або ж події це усі різні дії користувачів сайту, на які може реагувати веб-сторінка

Для того, щоб запустити код у той момент, коли документ готовий для того, щоб можна було маніпулювати його елементами, використовується вираз, відомий як івент document.ready().

Код на слайді створить обробник події готовності документа, який виведе повідомлення на екран в той момент, коли документ завантажиться.

Саме цей підхід з використанням jQuery є рекомендованим, на відміну від задання обробника події window.onload.

**В чому ж різниця між документ реді і віндов анлоуд?** Справа в тому що віндов анлоуд запускається після повного завантаження html-сторінки, тобто після завантаження контенту, включно з файлами мультимедіа типу фото і відео, скріптів, файлів css ну і всього іншого. Натомість метод документ реді запускається після завантаження дом дерева, тобто ігнорує завантаження фото та відео та іншого контенту. Якщо наш файл з кодом має великий розмір, то це допомагає нам значно пришвидшити написання коду, відкидаючи час очікування на віндов анлоуд. І наскільки мені відомо, віндов анлоуд працює нестабільно з деякими браузерами, тому все ж таки краще віддавати перевагу документ реді.

**10)** Далі поговоримо про зчитування та зміну вмісту HTML

Тут варто виділити три методи: це html, text i val

Метод .html() дозволяє зчитувати та записувати HTML-вміст елементів сторінки.

Коли цей метод використовується як геттер, він не приймає аргументів і повертає вміст першого елементу в наборі.

У наведеному прикладі у змінну htmlString записується вміст елемента сторінки з айдішкою 'mydiv'.

Коли метод використовується як сеттер, то він задає HTML-вміст всіх елементів в наборі. Тобто будь-який попередній вміст цих елементів повністю замінюється новим.

В нижньому прикладі відбувається заміна вмісту всіх div-елементів сторінки на абзац, що містить рядок 'Hello!'.

Метод text()не можна використовувати для введення даних форми або скріптів. Щоб встановити або отримати текстове значення input або textarea елементів, тому нам треба використовувати метод [.val()](https://api.jquery.com/val/).

Ну і починаючи з версії jQuery 1.4, метод .text()повертає значення вузлів text і CDATA, а також значення вузлів елементів.

**11)** Далі йде робота з класами

jQuery дозволяє читати, додавати, чи видаляти інформацію про клас для довільного елементу. Це може бути корисно для зміни того, як елементи будуть відображатися, виходячи з визначених CSS-стилів, назначених класу. Звичайно, існують інші ситуації, де це може буде корисно, отож розглянемо конкретні методи.

Першим методом є .addClass(), який дозволяє додати клас, назва якого передається як параметр

Наступним йде метод .removeClass(), який працює протилежним чином до попереднього, тобто видаляє клас

Третій метод це .hasClass() – *визначає, чи елементу присвоєно даний клас. Він повертає тру або фолс в залежності від результату роботи*

І, нарешті, метод .toggleClass() який додає клас, якщо його немає, чи, навпаки, видаляє, якщо він є.

Ці методи, як і інші методи jQuery, можуть об'єднуватися у ланцюг один до одного, як наведено у прикладі: спочатку за допомогою методу removeClass у всіх параграфів ми видаляємо класи 'myClass' та 'noClass', потім додаємо клас 'yourClass'. Такий запис коли ми викликаємо багато функцій одна після одної називається ланцюговими функціями

**ПОКАЗАТИ ФОРМУ**

**12) Ще трохи про івенти**

Бібліотека jQuery надає зручні можливості для встановлення обробників подій для елементів сторінки

Ці події, як правило, викликаються внаслідок реакції на дії кінцевого користувача сторінки таких як, наприклад, введення тексту в елементи форми, чи рух курсору миші.

В деяких випадках, наприклад, коли завантажується, чи вивантажується сторінка, сам браузер викликає подію.

jQuery надає методи для більшості рідних подій браузера, зокрема, .click(), .focus(), .blur(), .change().

**13)** Також хотів би трохи розказати про подію онклік

Наведений на екрані приклад встановлює обробник onclick для всіх абзаців на сторінці, для чого за допомогою селектора 'p' обираються всі абзаци, далі методу click() передається анонімна функція, у тілі якої декларується код, що буде виконуватися у випадку спрацювання обробника – у даному разі це виведення повідомлення у консоль.

Використання методу on() дозволяє задати обробник як для будь-якої стандартної події браузера, так і для нестандарних подій.

Наприклад, передавши у якості першого аргументу назву метода 'click', а далі анонімну функцію із обробником події, можна отримати таку ж функціональність, як і у випадку використання методу click(), який був розглянутий раніше.

Для обробки багатьох подій необхідно передати назви цих подій методу on(), розділяючи їх пробілами, як можна побачити у наведеному прикладі.

**14) Далі в мене про розмірності, але я так подумав що тут нема сенсу про це все розповідати, тому думаю перейду зразу до наступного слайду**

Метод width()встановлює або повертає ширину елемента (за винятком відступу, рамки та поля).

Метод height()встановлює або повертає висоту елемента (за винятком відступу, рамки та поля).

Метод innerWidth()повертає ширину елемента (включаючи відступи).

Метод innerHeight()повертає висоту елемента (включаючи відступи).

Метод outerWidth()повертає ширину елемента (включаючи відступ і рамку).

Метод outerHeight()повертає висоту елемента (включаючи відступ і рамку).

Метод outerWidth(true)повертає ширину елемента (включаючи відступи, межі та поля).

Метод outerHeight(true)повертає висоту елемента (включаючи відступи, межі та поля).

**15) Обхід джейквері**

Обхід jQuery використовується для «пошуку» елементів HTML на основі їх зв’язку з іншими елементами.

На скріні показано HTML-сторінку у вигляді дерева DOM. За допомогою треверсінгу (обходу) jQuery ми можемо легко рухатися вгору (попередники), вниз (нащадки) і вбік (побратими) у дереві, починаючи з вибраного елемента. Цей рух по дереві і називається обходом. Думаю розкажу кілька слів про ієрархію в самому дереві, щоб всім був зрозумілий принцим

* Елемент <div> є **батьком** <ul> і **предком** усього, що входить до нього
* Елемент <ul> є **батьківським** для обох елементів <li> і **дочірнім** для <div>
* Лівий елемент <li> є **батьківським** для <span>, **дочірнім** для <ul> і **нащадком** для <div>
* Елемент <span> є **дочірнім** елементом лівого <li> та **нащадком** <ul> і <div>
* Два елементи <li> є братами і **сестрами** (у них один батьківський елемент)
* Правий елемент <li> є **батьківським** для <b>, **дочірнім** для <ul> і **нащадком** для <div>
* Елемент <b> є **дочірнім** елементом правого <li> та **нащадком** <ul> і <div>

jQuery надає різноманітні методи, які дозволяють нам переглядати DOM, трохи детальніше в наступному слайді.

**16)** Обхід дерева ДОМ – Предки. Тут я виділив три методи: це parent, parents i parentsUntil.

Перший з них -  parent(), він повертає прямий батьківський елемент вибраного елемента.

Варто зазначити, що цей метод проходить лише один рівень дерева DOM, тобто він нам НЕ поверне всіх предків селектора, який ми вкажемо йому у якості параметра.

Наступний метод  - це parents(), ось він вже повертає всі елементи-предки вибраного елемента, аж до кореневого елемента документа ( <html>). Ми також можемо використовувати додатковий параметр для фільтрації пошуку предків, наприклад вказавши в дужках як аргумент «ul»

І останній метод parentsUntil() – він повертає всі елементи-предки між двома заданими аргументами.

**17)** Далі йде теж обхід ДОМ дерева, тільки по дочірніх елементах. Тут є два методи, це children та find.

Метод children() повертає всіх прямих дочірніх елементів вибраного елемента. Цей метод як і відповідний метод для предків проходить лише один рівень дерева DOM.

Тут ми також можемо використовувати додатковий параметр для фільтрації

Ну і метод find() повертає елементи-нащадки вибраного елемента, аж до останнього нащадка.

Ми також можемо знаходити братів і сестер елементів, тобто тих які мають один спільний батьківський елемент. Проте там трохи більше тих всяких методів, не думаю що це буде цікаво слухати, тому я їх просто перечислю без детального опису кожного. Це siblings()

* next()
* nextAll()
* nextUntil()
* prev()
* prevAll()
* prevUntil()

В принципі по назвах методів тут все інтуїтивно зрозуміло. А от про фільтрацію я хотів би розповісти трохи більше, отже

**18)** Найпростішими методами фільтрації є first(), last()і eq(), які дозволяють вибрати певний елемент на основі його позиції в групі елементів.

Інші методи фільтрації, наприклад filter()і not() дозволяють вибирати елементи, які відповідають або не відповідають певним критеріям.

**Трохи детальніше про самі методи:** Метод first()повертає перший елемент із зазначених елементів. Метод last() повертає відповідно останній елемент.

Метод eq() повертає елемент із певним номером індексу вибраних елементів. В нього до речі є цікава особливість: він може приймати звичайне число, тобто пошук буде відбуватись з початку, а може приймати параметр IndexFromEnd, який буде рахувати починаючи з кінця

Індексація тут починається як і очікувалось з 0, тому перший елемент матиме індекс 0.

Метод filter()дозволяє нам вказати критерій. Елементи, які не відповідають критеріям, будуть видалені з вибору, а ті, що відповідають будуть повернені.

Ну і метод not() повертає всі елементи, які не відповідають критеріям. І цей самий not() протилежний до методу filter().