Об’єктна модель браузера

План:

1. Загальна характеристика, призначення, та властивості об’єктів верхнього рівня.
2. Визначення об’єктів верхнього рівня та їх ролі в WEB-документах.
3. Роль таймера в WEB-документах та його призначення.
4. Процес створення та вбудовування таймера в WEB-документи.

Загальна характеристика, призначення, та властивості об’єктів верхнього рівня.

Об'єктна модель браузера (англ. Browser Object Model, BOM) є однією з ключових складових веб-браузера, що визначає інтерфейс для взаємодії між JavaScript-кодом та браузером. Вона надає доступ до об'єктів та методів, які представляють структуру та функціональні можливості браузера.

Основна мета об'єктної моделі браузера полягає в тому, щоб надати розробникам можливість маніпулювати та контролювати браузерні елементи з допомогою JavaScript. Вона дозволяє отримувати доступ до властивостей, методів та подій браузера, а також змінювати їх стан.

Загальна характеристика, призначення та властивості об’єктів верхнього рівня в об'єктній моделі браузера є важливими поняттями в розумінні того, як браузери взаємодіють зі сторінками веб-сайтів та іншими елементами веб-додатків.

Об’єктна модель браузера визначає спосіб представлення структури та поведінки веб-елементів, таких як документи, елементи керування, скрипти та інші, у вигляді об'єктів програмування.

Загальна характеристика, призначення та властивості об’єктів верхнього рівня в об’єктній моделі браузера дозволяють розробникам створювати потужні веб-додатки та веб-сторінки з динамічними функціями.

Визначення об’єктів верхнього рівня та їх ролі в WEB-документах.

Об'єкти верхнього рівня в об'єктній моделі браузера є глобальними об'єктами, які представляють основні компоненти та функціональні можливості браузера. Вони надають доступ до різних властивостей, методів та подій, що дозволяють взаємодіяти з браузерним середовищем та маніпулювати веб-документами.

Призначення об’єктів верхнього рівня полягає в наданні програмістам інтерфейсу для взаємодії з різними компонентами браузера та маніпулювання ними. Це дозволяє розробникам створювати динамічні веб-сторінки та веб-додатки, що реагують на дії користувача та забезпечують багатофункціональність.

Об’єктна модель браузера включає такі об’єкти верхнього рівня:

- Об’єкт "window": Це головний об’єкт браузера, який представляє вікно браузера або вкладку. Він містить глобальні методи та властивості, доступні в усіх скриптах на сторінці.
- Об’єкт "document": Цей об’єкт представляє HTML-документ, що відображається в поточному вікні браузера. Він надає доступ до вмісту сторінки та методи для маніпулювання елементами сторінки, наприклад, додавання або видалення елементів.
- Об’єкт "navigator": Цей об’єкт містить інформацію про браузер, що використовується, таку як назва, версія та підтримувані функції. Він також надає доступ до інформації про користувача, такої як мова та місцезнаходження.
- Об’єкт "XMLHttpRequest": Цей об’єкт використовується для виконання асинхронних HTTP-запитів до сервера. Він дозволяє веб-сторінці взаємодіяти з сервером без перезавантаження сторінки, наприклад, для завантаження даних з сервера або відправки даних на сервер.
- Об’єкт "Event": Цей об’єкт представляє подію, яка сталася на сторінці, наприклад, натискання кнопки миші або завантаження сторінки. Він містить інформацію про подію та методи для обробки подій.

Ці об’єкти надають доступ до багатьох можливостей браузера та дозволяють взаємодіяти з користувачем, маніпулювати сторінками та виконувати асинхронні операції з сервером.

Ці об’єкти верхнього рівня мають свої властивості та методи, які можуть бути використані для маніпулювання та взаємодії з іншими елементами браузера та сторінками.

Наприклад, об’єкт "document" має методи для доступу до елементів сторінки, зміни їх вмісту, стилів та властивостей. Об’єкт "window" надає методи для відкриття нових вікон, встановлення таймерів та керування розміром та положенням вікна браузера.

Роль таймера в WEB-документах та його призначення.

Таймер в WEB-документах є функціональним засобом, який дозволяє веб-розробникам планувати виконання певних функцій або блоків коду в певний час або з певним інтервалом. Він є однією з основних можливостей, які надає об'єктна модель браузера.

Роль таймера в WEB-документах та його призначення в об'єктній моделі браузера полягають у керуванні виконанням коду JavaScript в певний час або з певним інтервалом. Таймери дозволяють розробникам планувати виконання функцій або блоків коду відповідно до заданих умов часу.

Об'єктна модель браузера надає два основні методи для роботи з таймерами:

- setTimeout(): Цей метод дозволяє виконати певну функцію або блок коду один раз після зазначеного часового інтервалу. Синтаксис методу setTimeout() виглядає наступним чином:

setTimeout(function, delay);

Function представляє функцію або код, який потрібно виконати, а delay вказує затримку (в мілісекундах) перед виконанням.

Наприклад, такий код запускає функцію myFunction() через 3 секунди:

setTimeout(myFunction, 3000);

- setInterval(): Цей метод дозволяє виконувати певну функцію або блок коду регулярно зазначений інтервал часу. Синтаксис методу setInterval() виглядає так:

  setInterval(function, interval);

  Function вказує функцію або код, який повинен виконуватися, а interval вказує інтервал часу (в мілісекундах) між кожним виконанням.

  Наприклад, такий код буде виконувати функцію myFunction() кожні 2 секунди:

  setInterval(myFunction, 2000);

Таймери дозволяють виконувати різноманітні дії в WEB-документах, такі як анімація, періодичне оновлення даних, планування подій, затримки перед виконанням певних дій та багато іншого. Вони розширюють можливості динамічного та інтерактивного веб-дизайну, дозволяючи контролювати виконання коду відповідно до потреб розробника та вимог застосування.

Процес створення та вбудовування таймера в WEB-документи.

Процес створення та вбудовування таймера в WEB-документи в об'єктній моделі браузера включає кілька кроків, які дозволяють веб-розробникам контролювати часові аспекти виконання коду на стороні клієнта. Основні кроки включають наступне:

1. Створення функції або блоку коду: Спочатку необхідно визначити функцію або блок коду, який потрібно виконати після затримки або з певним інтервалом.

```
function myFunction() {
  console.log("Цей код буде виконано після затримки.");
}
```

2. Використання методу setTimeout() або setInterval(): Об'єкти window.setTimeout() або window.setInterval() надають можливість встановити таймер для виконання функції або блоку коду.

```
// Виконання коду один раз після 3-секундної затримки
setTimeout(myFunction, 3000);

// Виконання коду кожні 2 секунди
setInterval(myFunction, 2000);
```

3. Призначення таймера в об'єкті window: Після виклику методу setTimeout() або setInterval(), таймер створюється і прив'язується до об'єкту window. Таким чином, браузер починає відлік затримки або інтервалу часу.

```
// setTimeout
```

```
var timeoutId = setTimeout(myFunction, 3000);
// setInterval
var intervalId = setInterval(myFunction, 2000);
```

4. Обробка функції або блоку коду: Коли затримка або інтервал часу закінчується, браузер виконує функцію або блок коду, які були визначені для таймера.

Код, який був визначений у функції myFunction(), буде виконано після вказаної затримки або з певним інтервалом.

5. Зупинка таймера (за потреби): Якщо необхідно зупинити таймер до закінчення вказаного часу або інтервалу, можна використати метод clearTimeout() або clearInterval(), передавши йому ідентифікатор таймера.

```
// Зупинка setTimeout таймера
clearTimeout(timeoutId);
// Зупинка setInterval таймера
clearInterval(intervalId);
```

Вбудовання таймера в WEB-документ відбувається шляхом виклику методу setTimeout() або setInterval() у відповідному контексті, такому як подія клікання, завантаження сторінки або виконання певної умови. Таким чином, код таймера може бути активований у відповідь на певні події або в певні моменти виконання скрипту.

Цей процес дозволяє розробникам контролювати виконання коду веб-документа відповідно до потреб та вимог застосування, створювати

анімацію, оновлювати дані та створювати інтерактивні ефекти, що забезпечують більш гнучкий та персоналізований досвід для користувачів.