Контекст функції

№ уроку:	10	**Курс:**	JavaScript Базовий

Засоби навчання:	Visual Studio Code Web Browser

Огляд, мета та призначення уроку

Розглянути, що таке контекст функції та як він змінюється залежно від розташування або способу виклику функції. Навчитися прив'язувати контекст за допомогою методу bind. Вивчити способи планування запуску функцій у JavaScript.

Вивчивши матеріал цього заняття, учень зможе:

- Розуміти, яке значення в певному випадку розташоване в this функції, яка запускається.
- Використовувати методи call та apply.
- Прив'язувати контекст функції за допомогою методу bind.
- Використовувати методи setInterval та setTimeout для планування запуску функції.

Зміст уроку

1. Контекст функції.
2. Зміна контексту через apply та call.
3. Прив'язка контексту через метод bind.
4. Планування виклику функції.

Резюме

- **Контекст функції** – це значення ключового слова this. Залежить від розташування функції в коді та від способу запуску функції.
- Якщо функція визначена як глобальна (перебуває у тегу script), під час виклику такої функції контекстом функції є **глобальний об'єкт** (window, якщо код виконується у браузері).
- За увімкненого суворого режиму (use strict) глобальна функція не матиме контексту. Значення **this** дорівнюватиме **undefined**.
- Якщо функцію визначено як метод об'єкта, контекстом функції є цей об'єкт. Якщо функція буде скопійована в новий об'єкт, контекстом функції стане новий об'єкт.
- Якщо функція є конструктором, контекст встановлюється в новий об'єкт під час використання ключового слова **new**. Якщо використовується функція-конструктор і ключове слово new не вказувалося, контекстом буде об'єкт згідно з правилами, які описані вище. За використання **class** конструктор не може бути запущений без використання ключового слова **new**.
- Якщо використовується **getter** або **setter**, **this** вказує на об'єкт, в якому визначено конструкцію.
- Якщо **this** використовується у функції, яка розташована в ланцюжку прототипів, то **this** вказує на той об'єкт, на якому спочатку відбувся виклик, а не на прототип, в якому визначено функцію.

CyberBionic systematics

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com

- Якщо **this** використовується у функції обробника події DOM, **this** вказує на елемент, який ініціював подію, але ця поведінка може працювати не у всіх браузерах.
- Якщо під час запуску функції потрібно змінити контекст функції, використовуються методи **call** чи **apply**. Оскільки функція у JavaScript є об'єктом, то функція містить методи та властивості.

  function test() {}

  Для встановлення потрібного контексту під час запуску такої функції можна використовувати: test.call(context);

  test.apply(context);

  Відмінність між call та apply полягає у способі передання параметрів у функцію, для якої змінюється контекст. **Call** приймає параметри через кому, **apply** приймає параметри як масив. В іншому функції нічим не відрізняються. Використовується та, яка зручніша в конкретному випадку.
- Використовуючи метод **bind**, на функції можна створити нову функцію, але прив'язати до цієї функції вказаний контекст. Також можна прив'язати параметри, щоб під час виклику нової функції не потрібно було передавати значення для цих прив'язаних параметрів.
- **Стрілкові функції** не мають свого контексту та завжди використовують контекст оточення, у якому були визначені. Стрілкові функції ідеально підходять для функцій зворотного виклику. Якщо для функцій зворотного виклику використовуються звичайні функції, необхідно передбачити можливість зазначення контексту, якщо він використовується у функції.
- Для планування виклику функцій використовуються два глобальні методи: **setTimeout** і **setInterval**. **setTimeout** дає змогу запустити функцію після затримки один раз, а setInterval запускає функцію багаторазово через зазначений інтервал часу.
- Обидва методи повертають id таймера. Використовуючи clearTimeout і clearInterval і передаючи методи Id таймера, можна зупинити запланований виклик функції.

Закріплення матеріалу

- Яким буде контекст у глобальної функції?
- Який контекст у функції-конструктора, якщо виклик відбувся без ключового слова new?
- Який контекст у функції, яка визначена у прототипі?
- Які методи дають змогу змінити контекст функції, яка викликається?
- У чому різниця між call та apply?
- Які є методи планування виклику функцій, у чому їхня різниця?
- Як можна скасувати запланований виклик функції?

Самостійна діяльність учня

Виконайте завдання у директорії Tasks\013 Function Context. Текст завдань розташований у коментарях, у тегах script.

Рекомендовані ресурси

Ключове слово This
https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Operators/this

setTimeout
https://developer.mozilla.org/ru/docs/Web/API/WindowOrWorkerGlobalScope/setTimeout

setInterval
https://developer.mozilla.org/ru/docs/Web/API/WindowOrWorkerGlobalScope/setInterval

CyberBionic Systematics ®
19 Evgenia Sverstiuka Str., 5 floor
Kyiv, Ukraine

Tel.:0 800 337 146
E-mail: edu@cbsystematics.com
Site: www.edu.cbsystematics.com | itvdn.com