Java Script

NEXT

GENIUS space

План

Замикання

this

call

apply

bind

Замикання

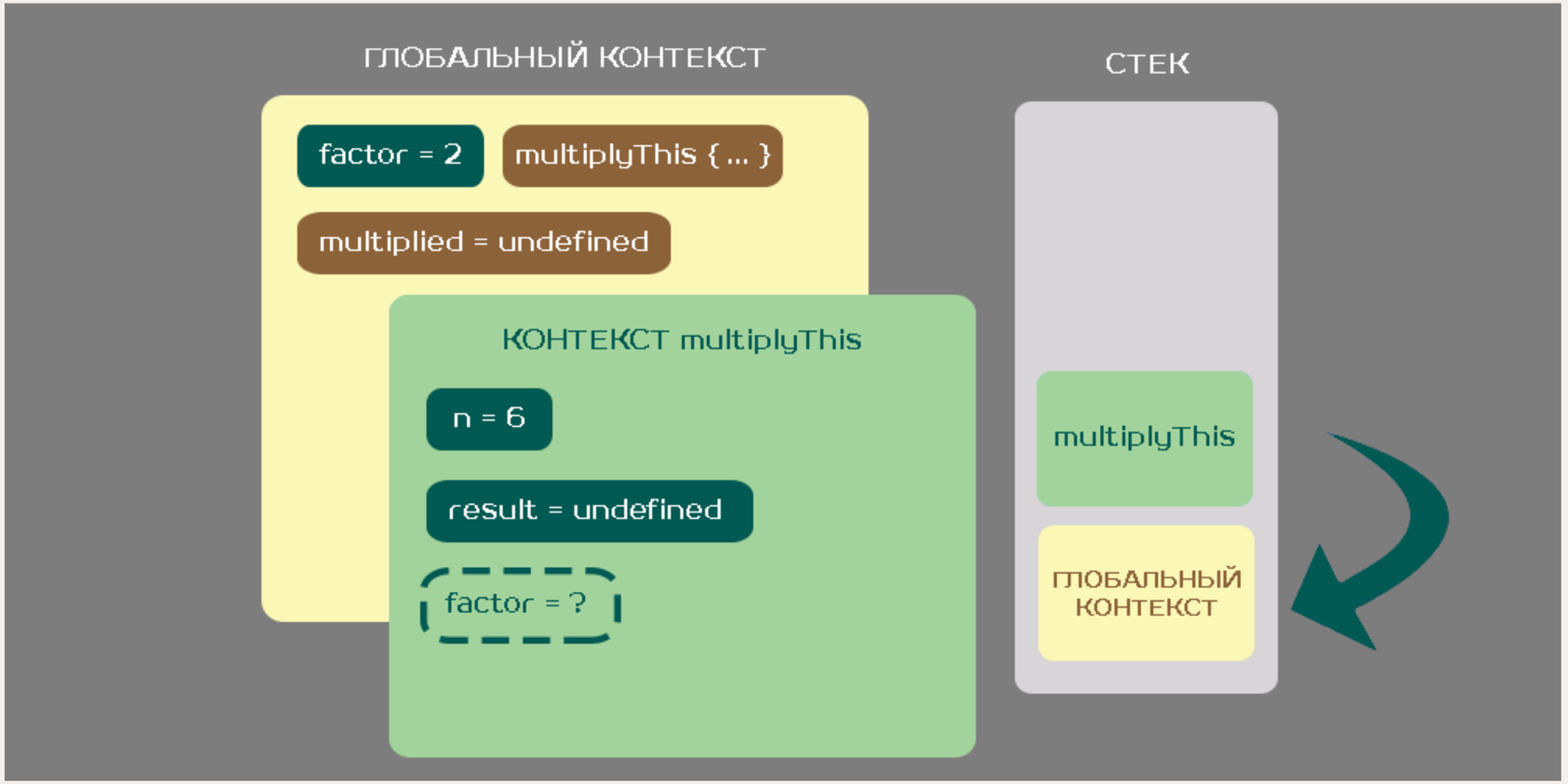
Замикання (closure) - це зв'язок між функцією та її ланцюжком областей видимості.

Цей механізм можливий завдяки лексичному оточенню. При виклику функції, її лексичне оточення отримує посилання ту область видимості, де функція було оголошено (зовнішнє лексичне оточення), цим отримуючи доступом до змінним із неї, буквально утримуючи у пам'яті таблицю доступних змінних.

Разом, замикання це термін який описує здатність функції запам'ятовувати лексичне оточення в якому вона була оголошена, і продовжувати отримувати доступ до змінних з цього лексичного оточення незалежно від того, де вона була викликана.

Тобто: функції автоматично запам’ятовують, де вони були створені, використовуючи приховану властивість [[Environment]], а потім їхній код може отримати доступ до зовнішніх змінних.

Коли під час співбесіди розробник отримує запитання “що таке замикання?”, правильною відповіддю буде визначення замикання та пояснення, що всі функції в JavaScript є замиканнями, і, можливо, ще кілька слів про технічні деталі: властивість [[Environment]], і як взагалі працюють лексичні середовища.



Ключове слово this є однією з найзаплутаніших частин JavaScript на старті вивчення.

Необхідно засвоїти лише одне правило для визначення this.
Значення контексту всередині функції визначаться над момент її створення, а момент виклику. Тобто значення цього визначається тим, як викликається функція, а не де вона була оголошена.

У глобальній області видимості, якщо скрипт виконується не в строгому режимі, this посилається на об'єкт window. У строгому режимі значення this, в глобальній області видимості, буде undefined.

Якщо функція була викликана як метод об'єкта, контекст буде посилатися на об'єкт, частиною якого є метод.

Методи об’єкту, "this"

Об’єкти зазвичай створюються для представлення сутностей реального світу, таких як користувачі, замовлення тощо:

```
let user = {
  name: "Іван",
  age: 30
};
```

І в реальному світі користувач може діяти: вибрати щось із кошика для покупок, авторизуватися, виходити із системи тощо.

Дії представлені в JavaScript функціями у властивостях об’єкта.

```
user.sayHi = function() {alert("Привіт!");}; user.sayHi(); // Привіт!
```

Як правило, метод об’єкта повинен отримувати доступ до інформації, що зберігається в об’єкті, для виконання своєї роботи.

Наприклад, коду всередині user.sayHi() може знадобитися ім’я, що зберігається в об’єкті user.

Для доступу до інформації всередині об’єкта метод може використовувати ключове слово this.

this у функціях зворотного виклику

Коли ми передаємо метод, який використовує this, як параметр, який буде використовуватися як функція зворотного виклику, буде проблема. Вирішення цієї проблеми розглядається в наступній секції.

```
const hotel = {
  name: 'Resort Hotel',
  showThis() {
    console.log(this);
  },
};
```

Під час виклику fn, callback буде посиланням на функцію showThis об'єкта hotel. Її виклик відбувається у глобальному контексті, про hotel вона нічого не знає. Відповідно this не посилатиметься на hotel

```
const fn = function(callback) {
  callback();
};
```

Передається посилання на функцію, а її виклик.

```
fn(hotel.showThis);
```

Стрілочні функції

Стрілочні функції не мають свого this. На відміну від звичайних функцій, змінити значення цієї всередині стрілки після її оголошення не можна.

Методи функцій call, apply, bind

Присвоєння функції як методу об'єкта може здатися гарною ідеєю. Але чи варто зберігати такі методи? Дублювання вже існуючих функцій як методів об'єкта займатиме ресурси, не приносячи жодних помітних вигод.

	call()	apply()	bind()
Execution	At the time of binding	At the time of binding	At the time when we execute the return function
Parameter	any number of arguments one by one.	Array []	array and any number of arguments
Is Return Function	Yes, it returns and calls the same function at that time of binding.	Yes, it returns and calls the same function at that time of binding.	Yes, it returns a new function or copy of the function. Which we can use whenever we want. It's like a loaded gun.

Запам'ятати правило використання call досить легко: метод call викличе функцію fnпередавши її це посилання на об'єкт obj, а також аргументи arg1, arg2 і т.д.

```
// const userInfo = {
//   name: "name",
//   age: 98,
//   logInfo: function (job) {
//     console.group(`${name} info:`);
//     console.log(`Name is : ${this.name}`);
//     console.log(`Age is : ${this.age}`);
//     console.log(`Job is : ${job}`);
//     console.groupEnd();
//   },
// };

// const Vano = {
//   name: "Ivan",
//   age: 45,
// };

// userInfo.logInfo.call(Vano, "developer");
```

Метод apply повний аналог методу call крім того, що синтаксис виклику аргументів вимагає не перерахування, а масив.

```
// const showUserInfo = {
//   name: name,
//   logInfo: function (job, city) {
//     console.group(`${name} info:`);
//     console.log(`Name is : ${this.name}`);
//     console.log(`Age is : ${this.age}`);
//     console.log(`Job is : ${job}`);
//     console.log(`City is : ${city}`);
//     console.groupEnd();
//   },
// };

// const Vano = {
//   name: "Ivan",
// };

// showUserInfo.logInfo.apply(Vano, ["developer", "Lviv"]);
```

Ми розглянули випадки, коли миттєво необхідно викликати функцію з іншим контекстом, для цього використовуються методи call і apply. Але у випадку функції зворотного дзвінка, коли необхідно не викликати функцію на місці, а передати посилання на цю функцію, причому з прив'язаним контекстом, call і apply не підходять. Метод bind дозволяє вирішити це завдання.

Метод bind створює копію функції fn з прив'язаним контекстом obj та аргументами arg1, arg2 тощо, після чого повертає її як результат своєї роботи. В результаті ми отримуємо копію функції з прив'язаним контекстом, яку можна передати куди завгодно і викликати будь-коли.

Найчастіше метод bind використовується для прив'язки контексту передачі методів об'єкта як функцій зворотного виклику. Візьмемо проблемний приклад із попередньої секції. Завдання прив'язки контесту ми можемо вирішити використовуючи метод bind, передавши функцією зворотного виклику копію методу з прив'язаним контекстом.

Додаткові матеріали

- https://uk.javascript.info/closure
- https://www.freecodecamp.org/news/lets-learn-javascript-closures-66feb44f6a44
- https://uk.javascript.info/object-methods
- https://www.freecodecamp.org/news/understand-call-apply-and-bind-in-javascript-with-examples/
- https://www.codingame.com/playgrounds/9799/learn-solve-call-apply-and-bind-methods-in-javascript
-

Домашнє завдання

--1--

1) написати об*єкт студента який буде виводити ім*я, спеціальнісь, середній бал і кількість пропущених занять

2) написати метод який буде виводити цю інформацію

3) написати три варіанти студентів

4) прикріпити зннaчення за допомогою call apply bind

--2--

Написати дві кнопки і закріпити на них функції

при натисканні на кнопку html - має видати коротке визначення що це таке

при натисканні на кнопку css - має видати коротке визначення що це таке

--3--

Написати функцію магазин, яка отримує назву товару, ціну за кг і кількість товару функція має повертати назву товару і вартість

перевірити на варіантах:

1) banana 30, 4,5

2) cherry 58,   1,3

3) jrange 89. 3,4