Модульность в JS. ESM.

Как правило, все крупные проекты имеют модульную структуру с целью инкапсуляции, многократного переиспользования и логического разделения кода на независимые блоки. Это позволяет подключать библиотеки, написанные другими разработчиками, а также упрощает отладку программ. Скорее всего Вы уже знакомы с тем, что такое модульность программ на примере других языков программирования.

До выхода стандарта ES2015 модули не было нативной поддержки модулей в JS. Для браузеров использовалась библиотека require.js, а в Node.js была реализована система CommonJS.

Теперь, начиная со стандарта ES2015, в JS появилась поддержка модулей (ESM – ECMAScript modules) и все современные браузеры, а также Node.js, тоже поддерживают ESM.

Модуль – это по сути один скрипт (файл с расширением .js или .mjs). Однако следует учесть, что расширение .mjs поддерживается не всеми инструментами (подробности в [4]), поэтому мы пока что пользуемся .js. Чтобы указать, что такой файл является именно модулем, а не просто скриптом, необходимо задать атрибут type:

```
<!doctype html>
<script type="module">
  import {sayHi} from './say.js';

  document.body.innerHTML = sayHi('John');
</script>
```

Каждый модуль имеет свою область видимости переменных и функций. Например, переменная «const variable», объявленная в двух разных модулях, не приведет к конфликту имён. Также каждый модуль по умолчанию устанавливает «strict» режим исполнения кода в нём, то есть директиву «use strict» можно не писать явно.

Обмен локальными переменными и функциями с другими модулями осуществляется посредством директив import (импорт объекта в модуль) и export (экспорт объекта из модуля). Необходимо отметить, что импорт нельзя осуществить в скрипт, не являющимся модулем, так как он не поддерживает директиву «import».

Директива «export» может быть указана как непосредственно перед экспортируемым объектом:

```
export const param = 10;

export const add = (a, b) => {
    return a + b;
};

export function mul(a, b) {
    return a * b;
}
```

так и для всех объектов скопом:

```
const param = 10;

const add = (a, b) => {
    return a + b;
};

function mul(a, b) {
    return a * b;
}

export { param, add, mul };
```

Импорт происходит поимённо (пример экспорта константы из ex.js в im.js):

```
//ex.js
export const param = 10;

//im.js
import { param } from "ex.js";

```

Существует также экспорт «по умолчанию», особенно удобный для экспорта одной сущности из модуля, хотя допускается комбинация экспорта по умолчанию и поимённых экспортов. Такой экспорт

отличается добавлением ключевого слова «default» после директивы и при импорте не требует фигурных скобок «{ }». Ещё одно отличие – экспортируемый объект может не иметь имя, а при импорте ему необходимо сразу задать псевдоним:

```
//ex.js
export default {
    INIT: 0,
    SUCCESS: 1,
    ERROR: 2
};

//im.js
import STATUS_CODES from "ex.js";
```

Здесь мы привели лишь малую часть того, что представляют собой ESM. Для того, чтобы лучше разобраться со всеми возможностями ESM, рекомендуется прочитать статьи [1] и [2] и разобрать все приводимые там примеры. Дополнительно, если есть желание, можно ознакомиться и динамическими импортами [3], но на практике они редко используются.

Вопросы и задания

1. Для чего нужен строгий режим strict mode в коде JS-скрипта? Какая директива используется для его установки и в каких областях видимости (scopes) может быть использована? Какие ограничения устанавливает строгий режим по сравнению с обычным режимом? Чему равно значение global this в ESM и с чем это связано?

2. Допустим, что у Вас в проекте есть модуль config.js, экспортирующий различные константы для конфигурации проекта. Пусть также модуле main.js Вам необходимо использовать все константы из модуля config.js. Укажите приемлемый синтаксис, с помощью которого можно импортировать все константы из config.js в main.js в одну строку.

3. Что такое реэкспорт и для чего он используется? Напишите синтаксис реэкспорта в полном и коротком варианте на примере

выдуманных модулей. Напишите то же самое для реэкспорта экспорта по умолчанию.

4. ESM являются отложенными (deferred) скриптами по умолчанию. Объясните, чем отличаются deferred-скрипты от обычных? Что даёт атрибут async в тэге <script>?

Ссылки

[1]. https://learn.javascript.ru/modules

[2]. http://learn.javascript.ru/import-export

[3]. https://learn.javascript.ru/modules-dynamic-imports

[4]. https://developer.mozilla.org/ru/docs/Web/JavaScript/Guide/Modules#%D0%B2%D0%B7%D0%B3%D0%BB%D1%8F%D0%B4_%D1%81%D0%BE_%D1%81%D1%82%D0%BE%D1%80%D0%BE%D0%BD%D1%8B_%E2%80%94_.mjs_%D0%BF%D1%80%D0%BE%D1%82%D0%B8%D0%B2_.js