Раздел
Модули
Навигация по уроку
Выражение import()
Комментарии
Поделиться
Редактировать на GitHub

Динамические импорты

Инструкции экспорта и импорта, которые мы рассматривали в предыдущей главе, называются «статическими». Синтаксис у них весьма простой и строгий.

Во-первых, мы не можем динамически задавать никакие из параметров `import`.

Путь к модулю должен быть строковым примитивом и не может быть вызовом функции. Вот так работать не будет:

```
1  import ... from getModuleName(); // Ошибка, должна быть
```

Во-вторых, мы не можем делать импорт в зависимости от условий или в процессе выполнения.

```
1  if(...) {
2    import ...; // Ошибка, запрещено
3  }
4
5  {
6    import ...; // Ошибка, мы не можем ставить импорт в б
7  }
```

Всё это следствие того, что цель директив `import/export` – задать костяк структуры кода. Благодаря им она может быть проанализирована, модули могут быть собраны в один файл специальными инструментами, а неиспользуемые экспорты удалены. Это возможно только благодаря тому, что всё статично.

Но как мы можем импортировать модуль динамически, по запросу?

Выражение import()

Выражение `import(module)` загружает модуль и возвращает промис, результатом которого становится объект модуля, содержащий все его экспорты.

Использовать его мы можем динамически в любом месте кода, например, так:

```
1  let modulePath = prompt("Какой модуль загружать?");
2
3  import(modulePath)
4    .then(obj => <объект модуля>)
5    .catch(err => <ошибка загрузки, например если нет так
```

Или если внутри асинхронной функции, то можно `let module = await import(modulePath)`.

Например, если у нас есть такой модуль `say.js`:

```
1  // 🔖 say.js
2  export function hi() {
3    alert(`Привет`);
4  }
5
6  export function bye() {
7    alert(`Пока`);
8  }
```

...То динамический импорт может выглядеть так:

```
let {hi, bye} = await import('./say.js');

hi();
bye();
```

А если в `say.js` указан экспорт по умолчанию:

```
// 🔌 say.js
export default function() {
  alert("Module loaded (export default)!");
}
```

...То для доступа к нему нам следует взять свойство `default` объекта модуля:

```
let obj = await import('./say.js');
let say = obj.default;
// или, одной строкой: let {default: say} = await impor

say();
```

Вот полный пример:

Результат say.js index.html

```
<!doctype html>
<script>
  async function load() {
    let say = await import('./say.js');
    say.hi(); // Привет!
    say.bye(); // Пока!
    say.default(); // Модуль загружен (экспорт по умолчанию)
  }
```

На заметку:

Динамический импорт работает в обычных скриптах, он не требует указания `script type="module"`.

На заметку:

Хотя `import()` и выглядит похоже на вызов функции, на самом деле это специальный синтаксис, так же, как, например, `super()`.

Так что мы не можем скопировать `import` в другую переменную или вызвать при помощи `.call/apply`. Это не функция.

Комментарии

перед тем как писать...