

Классы

Навигация по уроку

Синтаксис «class»

Что такое класс?

Не просто синтаксический сахар

Class Expression

Геттеры/сеттеры, другие сокращения

Свойства классов

Итого

Задачи (1)

Комментарии

Поделиться




Редактировать на GitHub

[🏠](#) → [Язык программирования JavaScript](#) → [Классы](#)  18-го февраля 2020

Класс: базовый синтаксис

В объектно-ориентированном программировании класс – это расширяемый шаблон кода для создания объектов, который устанавливает в них начальные значения (свойства) и реализацию поведения (методы).

 Википедия

На практике нам часто надо создавать много объектов одного вида, например пользователей, товары или что-то ещё.

Как мы уже знаем из главы [Конструкторы, создание объектов через "new"](#), с этим может помочь `new function`.

Но в современном JavaScript есть и более продвинутая конструкция «class», которая предоставляет новые возможности, полезные для объектно-ориентированного программирования.

Синтаксис «class»

Базовый синтаксис выглядит так:

```
1 class MyClass {
2   // методы класса
3   constructor() { ... }
4   method1() { ... }
5   method2() { ... }
6   method3() { ... }
7   ...
8 }
```

Затем используйте вызов `new MyClass()` для создания нового объекта со всеми перечисленными методами.

При этом автоматически вызывается метод `constructor()`, в нём мы можем инициализировать объект.

Например:

```
1 class User {
2
3   constructor(name) {
4     this.name = name;
5   }
6
7   sayHi() {
8     alert(this.name);
9   }
10 }
11
12 // Использование:
13 let user = new User("Иван");
14 user.sayHi();
```



Когда вызывается `new User("Иван")`:

1. Создаётся новый объект.



2. `constructor` запускается с заданным аргументом и сохраняет его в `this.name`.

...Затем можно вызывать на объекте методы, такие как `user.sayHi()`.

⚠ Методы в классе не разделяются запятой

Частая ошибка начинающих разработчиков – ставить запятую между методами класса, что приводит к синтаксической ошибке.

Синтаксис классов отличается от литералов объектов, не путайте их. Внутри классов запятые не требуются.

Что такое класс?

Итак, что же такое `class`? Это не полностью новая языковая сущность, как может показаться на первый взгляд.

Давайте развеем всю магию и посмотрим, что такое класс на самом деле. Это поможет в понимании многих сложных аспектов.

В JavaScript класс – это разновидность функции.

Взгляните:

```
1 class User {
2   constructor(name) { this.name = name; }
3   sayHi() { alert(this.name); }
4 }
5
6 // доказательство: User - это функция
7 alert(typeof User); // function
```



Вот что на самом деле делает конструкция `class User {...}`:



1. Создаёт функцию с именем `User`, которая становится результатом объявления класса. Код функции берётся из метода `constructor` (она будет пустой, если такого метода нет).
2. Сохраняет все методы, такие как `sayHi`, в `User.prototype`.

При вызове метода объекта `new User` он будет взят из прототипа, как описано в главе [F.prototype](#). Таким образом, объекты `new User` имеют доступ к методам класса.

На картинке показан результат объявления `class User`:



Можно проверить вышесказанное и при помощи кода:

```
1 class User {
2   constructor(name) { this.name = name; }
3   sayHi() { alert(this.name); }
4 }
5
6 // класс - это функция
7 alert(typeof User); // function
8
9 // ...или, если точнее, это метод constructor
10 alert(User === User.prototype.constructor); // true
11
12 // Методы находятся в User.prototype, например:
13 alert(User.prototype.sayHi); // alert(this.name);
14
15 // в прототипе ровно 2 метода
16 alert(Object.getOwnPropertyNames(User.prototype)); // c
```





Не просто синтаксический сахар

Иногда говорят, что `class` – это просто «синтаксический сахар» в JavaScript (синтаксис для улучшения читаемости кода, но не делающий ничего принципиально нового), потому что мы можем сделать всё то же самое без конструкции `class`:

```
1 // перепишем класс User на чистых функциях
2
3 // 1. Создаём функцию constructor
4 function User(name) {
5   this.name = name;
6 }
7 // каждый прототип функции имеет свойство constructor п
8 // поэтому нам нет необходимости его создавать
9
10 // 2. Добавляем метод в прототип
11 User.prototype.sayHi = function() {
12   alert(this.name);
13 };
14
15 // Использование:
16 let user = new User("Иван");
17 user.sayHi();
```

Результат этого кода очень похож. Поэтому, действительно, есть причины, по которым `class` можно считать синтаксическим сахаром для определения конструктора вместе с методами прототипа.

Однако есть важные отличия:

1. Во-первых, функция, созданная с помощью `class`, помечена специальным внутренним свойством `[[FunctionKind]]: "classConstructor"`. Поэтому это не совсем то же самое, что создавать её вручную.

В отличие от обычных функций, конструктор класса не может быть вызван без `new`:

```
1 class User {
2   constructor() {}
3 }
4
5 alert(typeof User); // function
6 User(); // Error: Class constructor User cannot be in
```

Кроме того, строковое представление конструктора класса в большинстве движков JavaScript начинается с «class ...»

```
1 class User {
2   constructor() {}
3 }
4
5 alert(User); // class User { ... }
```

2. Методы класса являются неперечислимыми. Определение класса устанавливает флаг `enumerable` в `false` для всех методов в `"prototype"`.

И это хорошо, так как если мы проходимся циклом `for...in` по объекту, то обычно мы не хотим при этом получать методы класса.

3. Классы всегда используют `use strict`. Весь код внутри класса автоматически находится в строгом режиме.

Также в дополнение к основной, описанной выше, функциональности, синтаксис `class` даёт ряд других интересных возможностей, с которыми



мы познакомимся чуть позже.

Class Expression

Как и функции, классы можно определять внутри другого выражения, передавать, возвращать, присваивать и т.д.

Пример Class Expression (по аналогии с Function Expression):

```
1 let User = class {
2   sayHi() {
3     alert("Привет");
4   }
5 };
```

Аналогично Named Function Expression, Class Expression может иметь имя.

Если у Class Expression есть имя, то оно видно только внутри класса:

```
1 // "Named Class Expression"
2 // (в спецификации нет такого термина, но происходящее
3 let User = class MyClass {
4   sayHi() {
5     alert(MyClass); // имя MyClass видно только внутри
6   }
7 };
8
9 new User().sayHi(); // работает, выводит определение My
10
11 alert(MyClass); // ошибка, имя MyClass не видно за пред
```

Мы даже можем динамически создавать классы «по запросу»:

```
1 function makeClass(phrase) {
2   // объявляем класс и возвращаем его
3   return class {
4     sayHi() {
5       alert(phrase);
6     };
7   };
8 }
9
10 // Создаём новый класс
11 let User = makeClass("Привет");
12
13 new User().sayHi(); // Привет
```

Геттеры/сеттеры, другие сокращения

Как и в литеральных объектах, в классах можно объявлять вычисляемые свойства, геттеры/сеттеры и т.д.

Вот пример `user.name`, реализованного с использованием `get/set`:

```
1 class User {
2
3   constructor(name) {
4     // вызывает сеттер
5     this.name = name;
6   }
7
8   get name() {
9     return this._name;
10  }
11
12  set name(value) {
13    if (value.length < 4) {
```

Раздел

Классы

Навигация по уроку

Синтаксис «class»

Что такое класс?

Не просто синтаксический сахар

Class Expression

Геттеры/сеттеры, другие сокращения

Свойства классов

Итого

Задачи (1)

Комментарии

Поделиться



Редактировать на GitHub



```
14     alert("Имя слишком короткое.");
15     return;
16   }
17   this._name = value;
18 }
19
20 }
21
22 let user = new User("Иван");
23 alert(user.name); // Иван
24
25 user = new User(""); // Имя слишком короткое.
```

При объявлении класса геттеры/сеттеры создаются на `User.prototype`, вот так:

```
1 Object.defineProperty(User.prototype, {
2   name: {
3     get() {
4       return this._name
5     },
6     set(name) {
7       // ...
8     }
9   }
10 });
```

Пример с вычисляемым свойством в скобках `[...]`:

```
1 class User {
2
3   ['say' + 'Hi']() {
4     alert("Привет");
5   }
6
7 }
8
9 new User().sayHi();
```

Свойства классов

 Старым браузерам может понадобиться полифил

Свойства классов добавлены в язык недавно.

В приведённом выше примере у класса `User` были только методы. Давайте добавим свойство:

```
1 class User {
2   name = "Аноним";
3
4   sayHi() {
5     alert(`Привет, ${this.name}!`);
6   }
7 }
8
9 new User().sayHi();
```

Свойство `name` не устанавливается в `User.prototype`. Вместо этого оно создаётся оператором `new` перед запуском конструктора, это именно свойство объекта.

Итого

Раздел

[Классы](#)

Навигация по уроку

Синтаксис «class»

Что такое класс?

Не просто синтаксический сахар

Class Expression

Геттеры/сеттеры, другие сокращения

Свойства классов

Итого

Задачи (1)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Базовый синтаксис для классов выглядит так:

```
1 class MyClass {
2   prop = value; // свойство
3   constructor(...) { // конструктор
4     // ...
5   }
6   method(...) {} // метод
7   get something(...) {} // геттер
8   set something(...) {} // сеттер
9   [Symbol.iterator]() {} // метод с вычисляемым именем
10  // ...
11 }
```

`MyClass` технически является функцией (той, которую мы определяем как `constructor`), в то время как методы, геттеры и сеттеры записываются в `MyClass.prototype`.

В следующих главах мы узнаем больше о классах, включая наследование и другие возможности.

✓ Задачи

Перепишите класс

важность: 5

Класс `Clock` написан в функциональном стиле. Перепишите его, используя современный синтаксис классов.

P.S. Часы тикают в консоли. Откройте её, чтобы посмотреть.

[Открыть песочницу для задачи.](#)



решение



Проводим [курсы по JavaScript и фреймворкам.](#) ✕

💬 Комментарии

перед тем как писать...