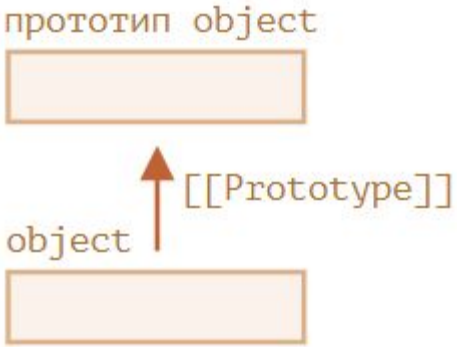
JavaScript: Prototypes, Classes

AIT
Hi-tech team

Часто в программировании мы часто хотим взять что-то и расширить.

Например, у нас есть объект **user** со своими свойствами и методами, и мы хотим создать объекты **admin** и **guest** как его слегка изменённые варианты. Мы хотели бы повторно использовать то, что есть у объекта **user**, не копировать/переопределять его методы, а просто создать новый объект на его основе.

В JavaScript объекты имеют специальное скрытое свойство **[[Prototype]]**, которое либо равно null, либо ссылается на другой объект. Этот объект называется **«прототип»**



Когда мы хотим прочитать свойство из object, а оно отсутствует, JavaScript автоматически берёт его из прототипа.

Прототип

Это внутренняя ссылка на другой объект

Для создания одного объекта на основе другого используется свойство **__proto__**

```
let animal = {
 eats: true,
 sleep() {
   console.log("Zzz-zzz-zz");
 },
};

let panda = {
 __proto__: animal,
};
```

Прототип

Важное замечание: отношение прототипного наследования - это **отношение между объектами**.

Если один объект имеет специальную ссылку __proto__ на другой объект, то при чтении свойства из него, если свойство отсутствует в самом объекте, оно ищется в объекте __proto__.

```
let animal = {
 eats: true,
 walk() {
   /* этот метод не будет использоваться в rabbit */
 }
};
let rabbit = {
 __proto__: animal
};

rabbit.walk = function() {
 alert("Rabbit! Bounce-bounce!");
};

rabbit.walk(); // Rabbit! Bounce-bounce!
```

Прототип

Свойство для задания прототипа

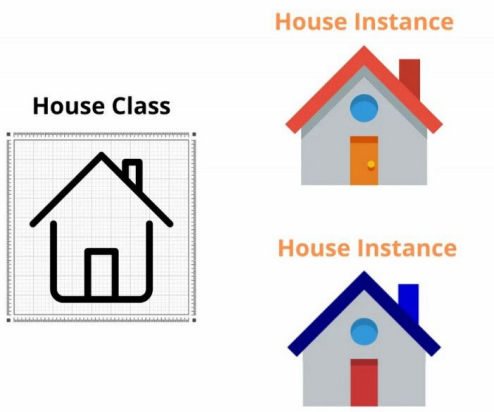
```
Object.setPrototypeOf(obj, prototype)
```

Свойство для проверки прототипа

```
Object.getPrototypeOf(obj)
```

Класс

Классы в JavaScript представляют собой шаблоны для создания объектов. Они предоставляют удобный способ определения объектов с общими свойствами и методами.



Создание класса

Для создания класса используется ключевое слово class. В классе можно определить конструктор и методы.

Конструктор - это специальный метод в классе, который вызывается при создании экземпляра объекта. Конструкторы используются для инициализации объекта, устанавливая начальные значения его свойств

```
// Объявили класс Wizard
class Wizard {
 constructor(name, house) {
   this.name = name;
   this.house = house;
 }
 introduce() {
   console.log(`I am ${this.name} from ${this.house}
house.`);
 }
}
// Создание экземпляра класса
const harry = new Wizard('Harry Potter', 'Gryffindor');
harry.introduce(); // "I am Harry Potter from Gryffindor
house."
```

EXTENDS

Наследование позволяет создавать новые классы, используя свойства и методы существующего класса.

Мы создали класс DarkWizard на основе класс Wizard

```
class DarkWizard extends Wizard {
 constructor(name, house, darkPower) {
   super(name, house);
   this.darkPower = darkPower;
 }

 useDarkPower() {
   console.log(`${this.name} uses dark power: ${this.darkPower}`);
 }
}

const voldemort = new DarkWizard('Lord Voldemort', 'Slytherin',
'Avada Kedavra');
voldemort.introduce(); // "I am Lord Voldemort from Slytherin
house."
voldemort.useDarkPower(); // "Lord Voldemort uses dark power:
Avada Kedavra"
```

Глобальный объект

Глобальный объект хранит переменные, которые должны быть доступны в любом месте программы.
Это включает в себя как встроенные объекты, например, **Array**, так и характерные для окружения свойства, например, **window.innerHeight** – высота окна браузера.

```
console.log(window);
```

```
// сработает только в браузере
alert("Привет"); // это то же самое, что и
window.alert("Привет");
```

Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH

AIT
Hi-tech team