

OOP

НАШИ ПРАВИЛА



Включенная камера



Вопросы по поднятой руке



Не перебиваем друг друга



Все вопросы, не связанные с тематикой курса (орг-вопросы и т. д.), должны быть направлены куратору



Подготовьте свое рабочее окружение для возможной демонстрации экрана (закройте лишние соцсети и прочие приложения)

ЦЕЛЬ

Изучить прототипно-ориентированную модель наследования

ПЛАН ЗАНЯТИЯ

- Прототип
- This
- Global object

Наследование

Что такое наследование (как вы его знаете по джаве)?

Object literal (литерал объекта)

Литерал объекта в JavaScript представляет собой способ создания объекта путем определения его свойств и их значений в фигурных скобках {}.

```
const person = {  
  name: 'John',  
  age: 25,  
  city: 'Berlin'  
};
```

Object literal (литерал объекта)

Объект в JavaScript - это составной тип данных, представленный коллекцией ключ-значение.

Ключи (также называемые свойствами) являются строками или символами, а значения могут быть любыми типами данных, включая другие объекты.

```
const person = {  
  name: 'John',  
  age: 25,  
  sayHello: function() {  
    console.log(`Hello, my name is  
${this.name}.`);  
  }  
};
```

Прототип

В плане наследования JavaScript работает лишь с одной сущностью: **объектами**.

Каждый объект имеет **внутреннюю ссылку** на другой объект, называемый его **прототипом**.

У объекта-прототипа также есть свой **собственный прототип** и так далее до тех пор, пока цепочка не завершится объектом, у которого свойство `prototype` равно `null`.

По определению, `null` не имеет прототипа и является завершающим звеном в цепочке прототипов.

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

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

```
panda.sleep(); // Zzz-zzz-zz
```


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!
```

THIS

Ключевое слово ``this`` в JavaScript используется для обращения к текущему объекту. Контекст ``this`` зависит от того, как вызывается функция.

```
const person = {  
  name: 'John',  
  introduce: function() {  
    console.log(`Hello, my name is ${this.name}.`);  
  }  
};  
  
person.introduce(); // "Hello, my name is John."
```

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(global); // если в node
// console.log(window); // если в браузере
```

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

THIS относится к одному из:

- **Глобальный объект:** В браузере это объект ``window``, в Node.js - ``global``. Содержит глобальные переменные и функции.
- **Текущий объект:** Это объект, к которому относится текущий контекст выполнения. В глобальной области видимости это глобальный объект.

CALL, APPLY

call и apply:

Методы используются для вызова функции с указанием конкретного объекта в качестве `this`. Разница между ними в передаче аргументов - `call` передает аргументы по одному, `apply` передает массив аргументов.

```
function greet(message) {  
  console.log(`${message}, ${this.name}.`);  
}  
  
const person = { name: 'John' };  
  
greet.call(person, 'Hello'); // "Hello, John."  
greet.apply(person, ['Hi']); // "Hi, John."
```

BIND

Метод **bind** создает новую функцию, привязывая указанный объект к `this` внутри функции.

```
function greet(message) {  
    console.log(`${message}, ${this.name}.`);  
}  
  
const person = { name: 'John' };  
  
const greetPerson = greet.bind(person);  
greetPerson('Hola'); // "Hola, John."
```

ПОИГРАЕМ ;)

■ Прототипное наследование
это отношение между чем и
чем?


■ Что такое `this`? На что он
указывает

■ Что делает метод `bind`?

ДОМАШНЕЕ ЗАДАНИЕ



СТАВИМ +, ЕСЛИ ВАМ ПОНЯТНО ДОМАШНЕЕ
ЗАДАНИЕ



Ваша новая IT-профессия – Ваш новый уровень жизни

Программирование с нуля в
немецкой школе AIT TR GmbH