پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم .

Constructor و عملگر new

@alithecodeguy

بوسیله گرامر {...} می‌توانیم یک object بسازیم ولی گاهی اوقات نیاز داریم تا چندین object مشابه بسازیم مثل کاربران متفاوت یا آیتم‌های مختلف یک منو یا ....
این کار می تواند توسط فانکشن constructor و عملگر new انجام شود .

فانکشن constructor

از نظر فنی ، فانکشن‌های constructor همان فانکشن‌های معمولی هستند . ولی دو چیز قراردادی را باید رعایت کنید :

۱– نام آنها با حرف بزرگ شروع می‌شود .

۲– فقط توسط عملگر new باید اجرا شوند .

برای مثال :

```
function User(name) {
  this.name = name;
  this.isAdmin = false;
}

let user = new User("Jack");

alert(user.name); // Jack
alert(user.isAdmin); // false
```

@alithecodeguy

وقتی فانکشنی بوسیله new اجرا می‌شود ، مراحل زیر اتفاق می‌افتد :

۱ . یک object خالی ساخته شده و در this قرار می گیرد .

۲ . بدنه فانکشن اجرا شده و معمولا this را تغییر داده و propertyهای جدید به آن اضافه می کند .

۳ . مقدار this برگردانده می‌شود .

Youtube , Linkedin , Telegram , Instagram : @alithecodeguy

پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم.

به عبارت دیگر ، عبارت new User(...) چنین کاری می‌کند :

```
function User(name) {
  // this = {};  (implicitly)

  // add properties to this
  this.name = name;
  this.isAdmin = false;

  // return this;  (implicitly)
}
```

@alithecodeguy

پس نتیجه مثال قبل ، object زیر می‌شود :

```
let user = {
  name: "Jack",
  isAdmin: false
};
```

حال اگر بخواهیم userهای دیگری بسازیم ، می‌توانیم به شکل زیر عمل کنیم :

```
new User("Ann")
new User("Alice")
```

این روش از روش عادی بسیار کوتاهتر بوده و خوانایی بیشتری نیز دارد . در واقع هدف اصلی constructor نیز همین است . ساخت objectهایی که قابلیت استفاده مجدد را دارند .

شاید بد نباشد که یک بار دیگر تذکر بدهیم که از نظر فنی ، از هر فانکشنی می‌توان به عنوان constructor استفاده کرد و این یعنی هر فانکشنی را بوسیله new می‌توانیم اجرا کنیم . حرف بزرگ ابتدایی در نام constructor در واقع قراردادی است که نشان می‌دهد فانکشن باید با new اجرا شود .

اگر کد ما فقط در مورد ساخت یک object بود ، می‌توانستیم به این صورت نیز عمل کنیم :

```
let user = new function() {
  this.name = "John";
  this.isAdmin = false;
  // ...other code for user creation
  // maybe complex logic and statements
};
```

@alithecodeguy

پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم .

در این حالت contructor نمی‌تواند جای دیگری استفاده شود چرا که جایی ذخیره نشده است . فقط ساخته و فراخوانی شده است . در واقع این روش برای این است که کدهای ساخت فقط یک object را encapsulate کرده بدون اینکه در آینده بخواهیم از آن استفاده کنیم .

Constructor mode test: new.target

مباحث این قسمت به ندرت استفاده شده و تنها در صورتی که علاقه‌مند هستید همه چیز را بدانید ، این بخش را مطالعه کنید .

بوسیله property مخصوص new.target درون یک فانکشن ، می‌توانیم متوجه بشویم که آن فانکشن ، عادی فراخوانی شده است یا با کلمه new اجرا شده است . (constructor mode)

برای فراخوانی‌های معمولی مقدار undefined برگردانده و برای فراخوانی‌هایی که با new انجام شده است ، مقدار function را برمی‌گرداند . مثال :

```
function User() {
  alert(new.target);
}

// without "new":
User(); // undefined

// with "new":
new User(); // function User { ... }
```

@alithecodeguy

همچنین می‌توانیم کاری کنیم تا هم فراخوانی‌های عادی و هم فراخوانی‌های با new ، کار یکسانی انجام دهند :

```
function User(name) {
  if (!new.target) { // if you run me without new
    return new User(name); // ...I will add new for you
  }
  this.name = name;
}

let john = User("John"); // redirects call to new User
alert(john.name); // John
```

@alithecodeguy

Youtube , Linkedin , Telegram , Instagram : @alithecodeguy

پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم.

گاهی اوقات از این روش در کتابخانه‌ها برای انعطاف پذیری بیشتر کدهای نوشته شده استفاده می‌شود. پس سایر برنامه نویسان می‌توانند فانکشن ها را با new یا بدون آن فراخوانی کنند و کد همچنان کار کند. اگرچه انجام این کار در همه جا درست نیست چرا که حذف new از خوانایی کد کم می‌کند. همه ما هرجایی که new ببینیم ، راحت متوجه می‌شویم که شی جدیدی ساخته شده است.

مقادیر بازگشتی constructorها

معمولا فانکشن‌های constructor مقدار بازگشتی ندارند. وظیفه آنها نوشتن تمام اطلاعات مورد نیاز روی this است که آن هم به صورت اتوماتیک به عنوان نتیجه در نظر گرفته می‌شود. ولی اگر بخواهید از return استفاده کنید ، باید یک قانون ساده را در نظر داشته باشید :

۱– اگر return ، یک object را برگرداند ، مقدار object به جای this در نظر گرفته می‌شود.

۲– اگر return ،یک primitive را برگرداند ، نادیده گرفته می‌شود.

به عبارت دیگر اگر return یک object برگرداند از آن استفاده می‌شود در غیر اینصورت نادیده گرفته می‌شود و از مقدار this استفاده می‌شود.

در مثال زیر از objectای که return برمی‌گرداند به جای this استفاده می‌شود:

```
function BigUser() {

  this.name = "John";

  return { name: "Godzilla" };  // <-- returns this object
}

alert( new BigUser().name );  // Godzilla, got that object
```

@alithecodeguy

و در مثال زیر return یک مقدار خالی را برمی‌گرداند :

```
function SmallUser() {

  this.name = "John";

  return; // <-- returns this
}
alert( new SmallUser().name );  // John
```

Youtube , Linkedin , Telegram , Instagram : @alithecodeguy

پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم .

معمولا constructorها return ندارند . در اینجا ما در مورد رفتار خاصی صحبت کردیم تا در مورد همه جنبه‌های موضوع این بخش صحبت کرده باشیم .

حذف پرانتزها

به عنوان یک نکته جانبی درنظر داشته باشید که اگر آرگومانی به constructor ارسال نمی‌کنیم ، می‌توانیم پرانتزهای جلوی آن را ننویسیم :

```
let user = new User; // <-- no parentheses
// same as
let user = new User();
```

حذف پرانتزها ، حرکت خوبی نیست ولی خواستیم نشان بدهیم که specification اجازه این کار را می‌دهد .

@alithecodeguy

methodها در constructor

استفاده از constructorها برای ساخت objectها انعطاف‌پذیری خوبی در اختیار ما قرار می‌دهد . فانکشن‌های constructor ممکن است پارامترهایی داشته باشند که نشان دهد چگونه object باید ساخته شود و چه چیزی درون آن باید قرار گیرد .

البته به جز propertyهای مختلف ، methodهایی نیز می‌توانیم به this اضافه کنیم .

در مثال زیر ، new User(name) یک object با name داده شده و متد sayHi ایجاد می‌کند .

```
function User(name) {
  this.name = name;
  this.sayHi = function() {
    alert( "My name is: " + this.name );
  };
}
let john = new User("John");

john.sayHi(); // My name is: John
/*
john = {
   name: "John",
   sayHi: function() { ... }
} */
```

@alithecodeguy

برای ساخت objectهای پیچیده‌تر ، syntaxهای پیچیده‌تری وجود دارد به نام class که بعدا در مورد آن صحبت خواهیم نمود .

Youtube , Linkedin , Telegram , Instagram : @alithecodeguy

پیشاپیش از هرگونه حمایت شما از جمله subscribe درکانال یوتیوب و کانال تلگرام سپاسگزارم.

خلاصه

- فانکشن‌های constructor یا به طور خلاصه constructorها ، فانکشن‌های معمولی هستند با اینکه تفاوت که به صورت قراردادی نام آن را به صورت PascalCase می‌نویسند .
- فانکشن‌های constructor می‌بایست با new فراخوانده شده و استفاده شوند . چنین فراخوانی‌ای باعث می‌شود که در ابتدا یک this خالی ایجاد شده و سپس و در نهایت this تکمیل شده بازگردانده می‌شود .

از فانکشن‌های constructor می‌توانیم برای ساخت چندین object مشابه استفاده کنیم .

جاوا اسکریپت از constructorها برای خیلی از objectهای داخلی زبان استفاده می‌کند مانند : Date ، Set و ... که در فصل‌های بعد در مورد آنها صحبت خواهیم کرد .

@alithecodeguy

Youtube , Linkedin , Telegram , Instagram : @alithecodeguy