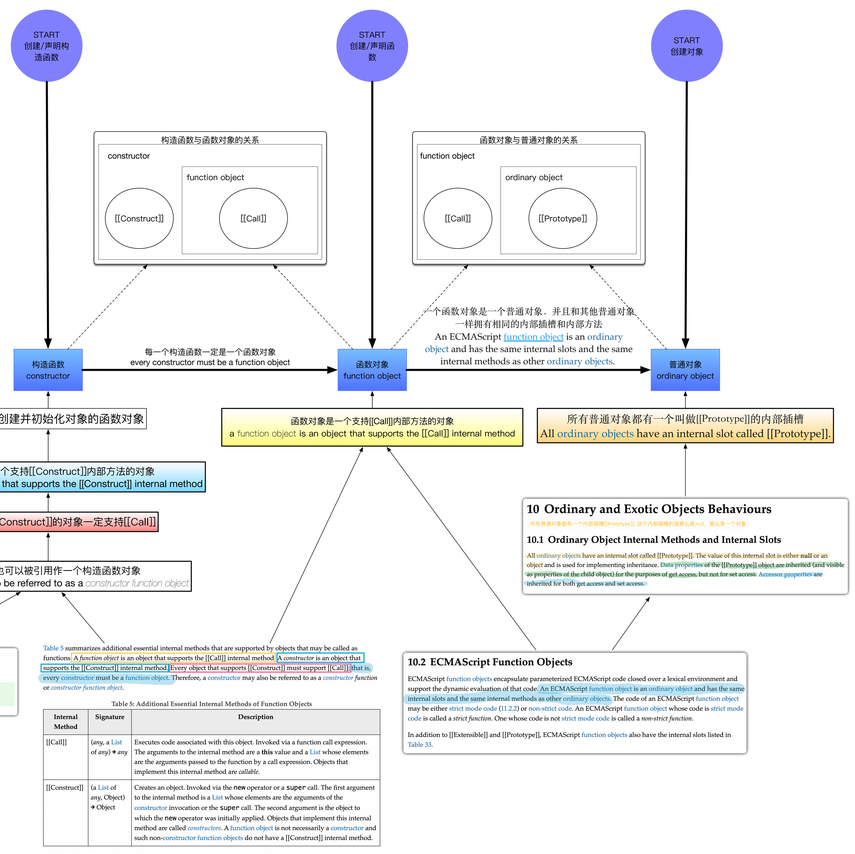
在程序员眼中，构造函数、函数、对象，它们是不同的东西，它们的联系是谁创造了谁的关系；
在JS眼中，构造函数、函数、对象，要想描述、区别、使用它们，就把它们看作为对象进行封装；

如果说一个对象是一个矩形，那么函数就是一个圆角矩形，而构造函数是一个更大的圆角矩形。
什么意思呢？也就是说，它们本质上都是矩形（都是对象），圆角矩形和更大圆角矩形的本质也是相同的圆角矩形（函数对象）

探求构造函数的本质



4.4.7 constructor

function object that creates and initializes objects
创建并初始化对象的函数对象

NOTE The value of a constructor's **"prototype"** property is a prototype object that is used to implement inheritance and shared properties.
构造函数的"prototype"属性是一个原型对象（prototype object），被用来实现继承和共享属性的。

10 Ordinary and Exotic Objects Behaviours

所有普通对象都有一个内部插槽叫[[Prototype]]，这个内部插槽的值要么是null，要么是一个对象

10.1 Ordinary Object Internal Methods and Internal Slots

All ordinary objects have an internal slot called [[Prototype]]. The value of this internal slot is either **null** or an object and is used for implementing inheritance. ~~Data properties of the [[Prototype]] object are inherited (and visible as properties of the child object) for the purposes of get access, but not for set access. Accessor properties are inherited for both get access and set access.~~

Table 5 summarizes additional essential internal methods that are supported by objects that may be called as functions. A function *object* is an object that supports the [[Call]] internal method. A *constructor* is an object that supports the [[Construct]] internal method. Every object that supports [[Construct]] must support [[Call]]; that is, every constructor must be a function object. Therefore, a constructor may also be referred to as a *constructor function* or *constructor function object*.

Table 5: Additional Essential Internal Methods of Function Objects

Internal Method	Signature	Description
[[Call]]	(*any*, a List of *any*) → *any*	Executes code associated with this object. Invoked via a function call expression. The arguments to the internal method are a **this** value and a List whose elements are the arguments passed to the function by a call expression. Objects that implement this internal method are *callable*.
[[Construct]]	(a List of *any*, Object) → Object	Creates an object. Invoked via the **new** operator or a **super** call. The first argument to the internal method is a List whose elements are the arguments of the constructor invocation or the **super** call. The second argument is the object to which the **new** operator was initially applied. Objects that implement this internal method are called *constructors*. A function object is not necessarily a constructor and such non-constructor function objects do not have a [[Construct]] internal method.

The semantics of the essential internal methods for ordinary objects and standard exotic objects are specified in clause 10. If any specified use of an internal method of an exotic object is not supported by an implementation, that usage must throw a **TypeError** exception when attempted.

10.2 ECMAScript Function Objects

ECMAScript function objects encapsulate parameterized ECMAScript code closed over a lexical environment and support the dynamic evaluation of that code. An ECMAScript function object is an ordinary object and has the same internal slots and the same internal methods as other ordinary objects. The code of an ECMAScript function object may be either strict mode code (11.2.2) or non-strict code. An ECMAScript function object whose code is strict mode code is called a *strict function*. One whose code is not strict mode code is called a *non-strict function*.

In addition to [[Extensible]] and [[Prototype]], ECMAScript function objects also have the internal slots listed in Table 33.

探求原型的起源

我们都知道，使用构造函数创建对象时，新创建对象的原型是指向构造函数的原型（不是构造函数本身）

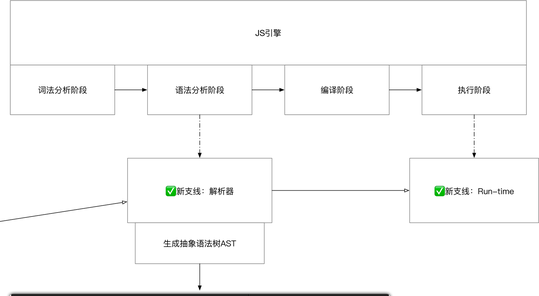
—>那么问题就落到了：构造函数的原型是谁装配的？
我们知道，构造函数是函数的一种，而函数定义时会被解析器预先处理

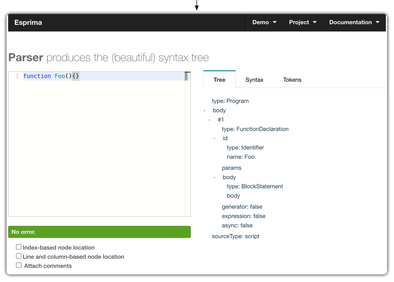
—>那么问题就落到了：解析器在遇到（构造）函数发生了什么？
解析器的作用只是根据源代码生成了一个相对静态的数据结构——AST，（构造）函数在解析器中也只是一个子树而已

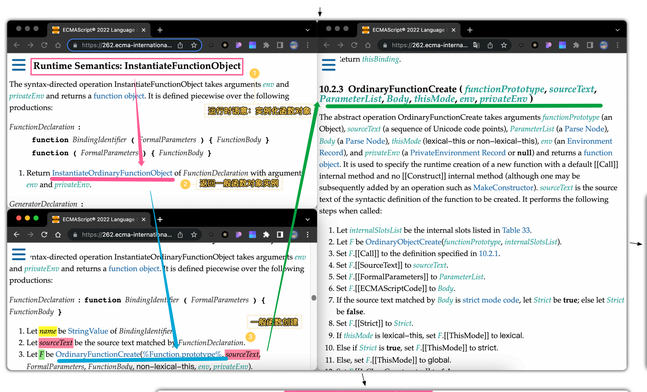
—>那么问题就落到了：解析器之后，函数是被谁？如何被处理的？
经过解析器，AST会被编译器编译成IR字节码，然后交由解释器转换为机器码，此后便由CPU进行执行

—>那么问题就落到了：执行代码期间，是怎么执行的？
根据ES规范，在运行时：
1. 首先识别到“声明函数”，然后执行抽象操作“InstantiateOrdinaryFunctionObject”；
2. 该操作中会执行一个“OrdinaryFunctionCreate()”方法，同时传入参数“%Function.prototype%”；
3. 在上面的方法中，会执行一步操作——“2. Let F be OrdinaryObjectCreate(functionPrototype, internalSlotsList)”；
4.在上面的操作执行方法中，会进一步执行——“Set O.[[prototype]] to proto”；
由此可知：**（构造）函数的原型，是在步骤1中执行的步骤2进行设置的**

—>那么问题就落到了：%Function.prototype%是什么？







10.1.12 OrdinaryObjectCreate (*proto* [, *additionalInternalSlotsList*])

The abstract operation OrdinaryObjectCreate takes argument *proto* (an Object or **null**) and optional argument *additionalInternalSlotsList* (a List of names of internal slots) and returns an Object. It is used to specify the runtime creation of new ordinary objects. *additionalInternalSlotsList* contains the names of additional internal slots that must be defined as part of the object, beyond [[Prototype]] and [[Extensible]]. If *additionalInternalSlotsList* is not provided, a new empty List is used. It performs the following steps when called:

1. Let *internalSlotsList* be « [[Prototype]], [[Extensible]] ».
2. If *additionalInternalSlotsList* is present, append each of its elements to *internalSlotsList*.
3. Let *O* be MakeBasicObject(*internalSlotsList*).
4. Set O.[[Prototype]] to *proto*.
5. Return *O*.

20.2.3 Properties of the Function Prototype Object

这里解释了 %Function.prototype%是什么的问题？

The *Function prototype object*:

- is *%Function.prototype%*.
- is itself a built-in function object.
- accepts any arguments and returns **undefined** when invoked.
- does not have a [[Construct]] internal method; it cannot be used as a constructor with the **new** operator.
- has a [[Prototype]] internal slot whose value is %Object.prototype%.
- does not have a **"prototype"** property.
- has a **"length"** property whose value is $+0_{\mathbb{F}}$.
- has a **"name"** property whose value is the empty String.

NOTE The Function prototype object is specified to be a function object to ensure compatibility with ECMAScript code that was created prior to the ECMAScript 2015 specification.

The *Function prototype object*:

- ~~is *%Function.prototype%*.~~ Function prototype object 就是所谓的Function.prototype
- ~~is itself a built-in function object.~~ Function prototype object 是它自己的一个内建函数对象
- accepts any arguments and returns **undefined** when invoked.
- does not have a [[Construct]] internal method; it cannot be used as a constructor with the **new** operator.
- ~~has a [[Prototype]] internal slot whose value is %Object.prototype%.~~ Function prototype object 有[[Prototype]]，指向 Object 的原型对象
- ~~does not have a **"prototype"** property.~~
- has a **"length"** property whose value is $+0_{\mathbb{F}}$.
- has a **"name"** property whose value is the empty String.

NOTE The Function prototype object is specified to be a function object to ensure compatibility with ECMAScript code that was created prior to the ECMAScript 2015 specification.

The Object constructor:

- is *%Object%*.
- is the initial value of the **"Object"** property of the global object.
- creates a new ordinary object when called as a constructor.
- performs a type conversion when called as a function rather than as a constructor.
- may be used as the value of an `extends` clause of a class definition.

20.1.2 Properties of the Object Constructor

The Object constructor:

- has a [[Prototype]] internal slot whose value is %Function.prototype%.
- has a **"length"** property.
- has the following additional properties:

Function很特殊，它的prototype属性和[[Prototype]]内部插槽都指向了一个对象——即Function.prototype
Function.prototype所代表的对象也很特殊，因为它没有独立的名字，只能靠Function的属性和内部插槽指代

之所以这样设计Function.prototype所指代的对象，是为了兼容ES6之前的版本

20.1.2.20 Object.prototype

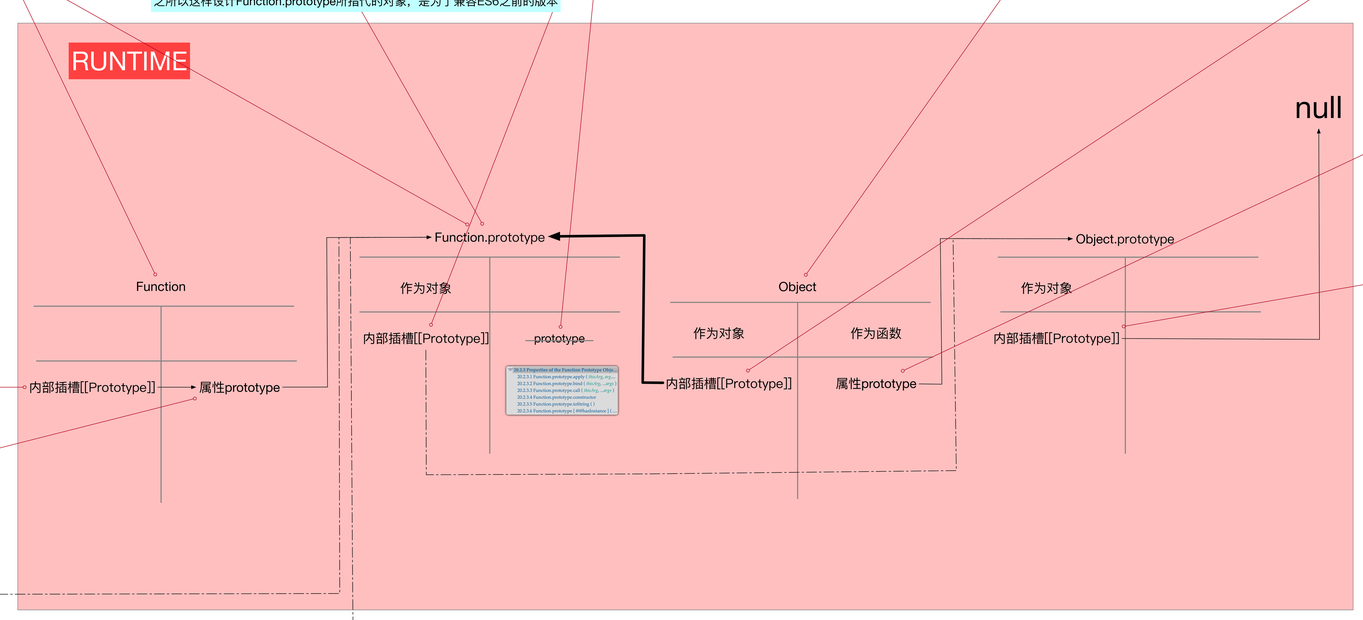
The initial value of `Object.prototype` is the Object prototype object.

This property has the attributes [[[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]: **false**].

20.1.3 Properties of the Object Prototype Object

The *Object prototype object*:

- is *%Object.prototype%*.
- has an [[Extensible]] internal slot whose value is **true**.
- has the internal methods defined for ordinary objects, except for the [[SetPrototypeOf]] method, which is as defined in 10.4.7.1. (Thus, it is an immutable prototype exotic object.)
- has a [[Prototype]] internal slot whose value is **null**.



The Function constructor:

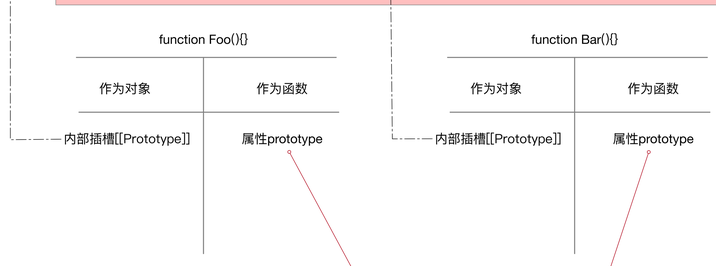
- is itself a built-in function object.
- has a [[Prototype]] internal slot whose value is %Function.prototype%.
- has the following properties:

20.2.2.2 Function.prototype

The value of `Function.prototype` is the Function prototype object.

This property has the attributes { [[Writable]]: **false**, [[Enumerable]]: **false**, [[Configurable]]: **false** }.

Function.prototype可以添加属性，但是不能更换该属性所绑定的对象



20.2.4.3 prototype 被当作构造函数使用的函数实例，都会有一个“prototype”属性。

Function instances that can be used as a constructor have a **"prototype"** property. Whenever such a Function instance is created another ordinary object is also created and is the initial value of the function's **"prototype"** property. Unless otherwise specified, the value of the **"prototype"** property is used to initialize the [[Prototype]] internal slot of the object created when that function is invoked as a constructor.

This property has the attributes { [[Writable]]: **true**, [[Enumerable]]: **false**, [[Configurable]]: **false** }.

NOTE Function objects created using `Function.prototype.bind`, or by evaluating a *MethodDefinition* (that is not a *GeneratorMethod* or *AsyncGeneratorMethod*) or an *ArrowFunction* do not have a **"prototype"** property.