Day2 - JS

Functions
Types of functions
Function statement vs Expression
Function Prototype
Execution Stack
Modules

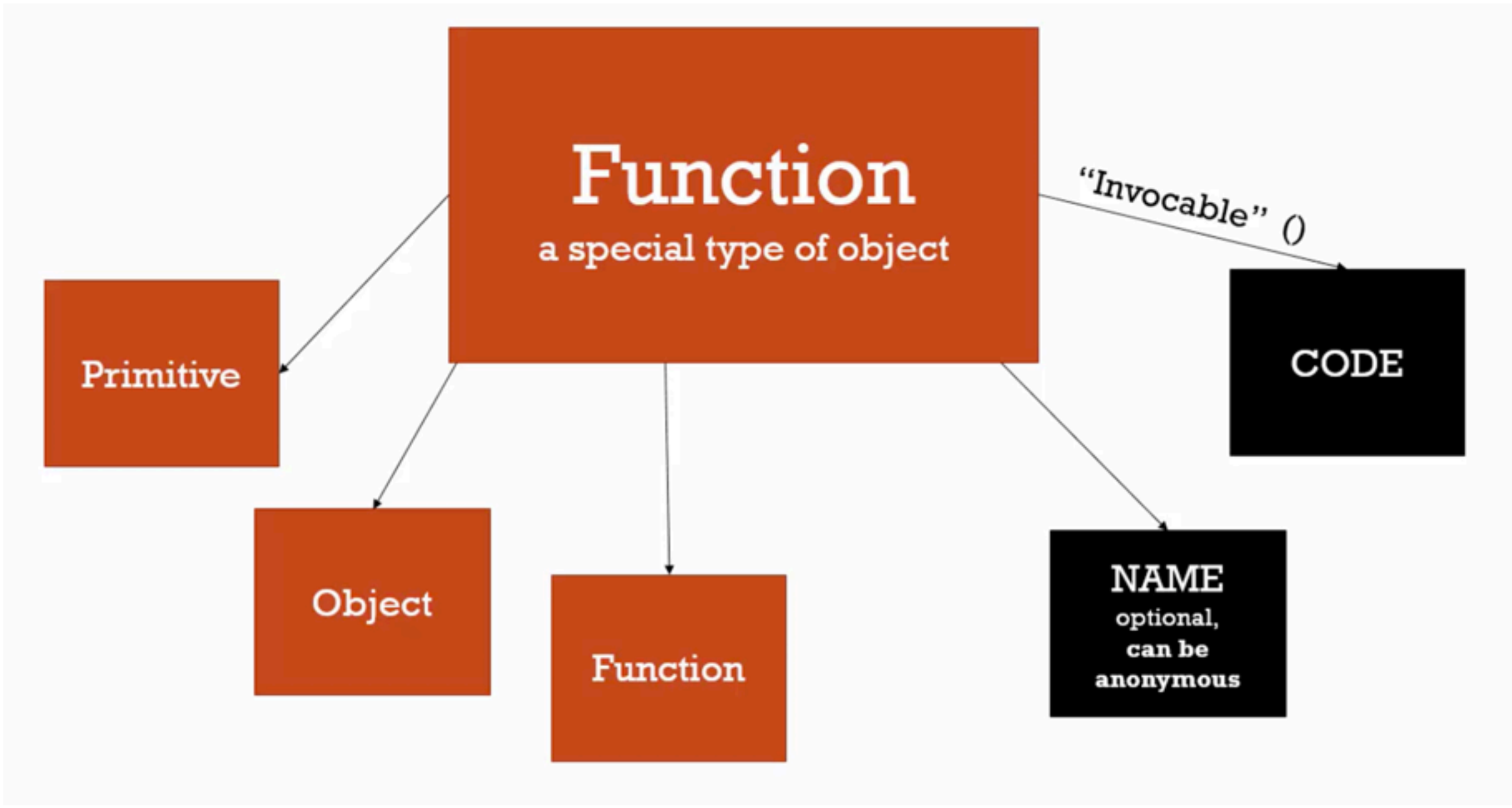
Js - Functions

- In javascript functions are objects

- Functions are first class citizens and can be used like any other object including
  - Assigned to a variable
  - Passed as an argument
  - Created on the fly
  - Returned from another function
  - Can also be part of an array

```
function greet() {
  console.log('hi');
}
// value can be assigned to function just like in objects
greet.language = 'English';
console.log(greet);
console.log(greet.language);  // output's English
```

Functions are objects with an invocable code



Types of Functions

- Named function
- Anonymous function - called only after definition
- Arrow Function -
- Immediately invoked function expression (IIFE)- save time travelling down the execution stack

Function Statement vs Function expression

- If the first token in the statement while parsing is a function then it is a function statement

    - Function statements are hoisted during syntax parsing

- If a function is assigned to a variable, it becomes a function expression

    - Function expressions are not hoisted, hence can be invoked only after declaration

- IIFE can be created and invoked on the fly, it is still an expression.

Function Prototype

- When a function is created in JavaScript, JavaScript engine adds a special property to the function called **prototype**

- Prototype property is an object (called as **prototype object**) which has a constructor (could be considered like an initialize in ruby Class) property by default

- The protoype object can be updated with some properties, which can be **accessed** by **all the objects created from this function**

- Note : In Js **instance method**

  can be created from function

```
function Human(firstName, lastName) {
  this.firstName = firstName,
  this.lastName = lastName,
  this.fullName = function() {
    return this.firstName + " " + this.lastName;
  }
}
var person = new Human("nandini", "nayak");
```

  Note : Both Js functions and classes contain prototype object

  Note: instance methods can access prototype properties but cannot update, only class or a function can update.

Execution Stack

```
function b() {
}

function a() {
      b();
}

a();
```

`this` concept in js

- `**this**` is a special variable created upon the creation of execution context and at the global level point to the global object - which is a **window object** in case of browser.

- Inside an object `**this**` variable refers to the object it resides in.

- ‘this’ keyword object in global functions refer to window object,

- ‘this’ keyword in methods(functions) in an object literal refers to corresponding object,

Modules

- Group similar functions in one file and export to another file
- These functions can be imported and used anywhere

```
module.exports.welcome = greet;

function greet() {
  return "hello";
}
// multiple function exported from module
module.exports = {
  hello: function() {
    return "hello";
  },
  hi: function() {
    return "hi"
  }
}
```

```
const myModule = require('./welcome');
//one function exported from the module
console.log(myModule.welcome());
// multiple functions exported from module
console.log(myModule.hello());
console.log(myModule.hi());
```