Explain the use of: "use strict"

It is not a statement, but a literal expression, ignored by earlier versions of JavaScript.
The purpose of "use strict" is to indicate that the code should be executed in "strict mode".
With strict mode, you cannot, for example, use undeclared variables.

Example:
See **UseStrict.js**

Variable/function-Hoisting

Variable-hoisting
JS is a bit different than other languages, which means that it can detect whether a variable just hasn't been declared yet or not at all. JS "looks ahead" through the rest of the code, to see if the variable in question has been declared.
If the variable isn't declared at all it will output "variableName is not defined".
If the variable hasn't been declared **yet** it will output "variableName: undefined".

Function-hoisting:
A function declaration doesn't just hoist the function's name. It also hoists the actual function definition. So you call a function before it's declared. However, **function definition hoisting only occurs for function declarations**, not function expressions.

Example:
See **Hoisting.js**

Explain >this< in JavaScript and how it differs from what we know from Java/.net.

This is a bit tricky in JavaScripts, but here goes:

- Whenever a function is contained in the global scope, the value of this inside of that function will be the global object (window in a browser) or undefined if in strict mode.
- Whenever a function is called by a preceding dot, the object before that dot is this.
- Whenever a constructor function is used, this refers to the specific instance of the object that is created and returned by the constructor function.
- Whenever JavaScript's call or apply method is used, this is explicitly defined.

Example of >this< used in a constructor function:
See **Object.js**

Function Closures and the JavaScript Module Pattern

A closure is a special kind of object that combines two things:

- A function.
- The environment in which that function was created. The environment consists of any local variables that were in-scope at the time that the closure was created.
-

JavaScript does not, like Java, provide a native way of providing private methods. But it is possible to emulate this using closures.

Example:
See **ReusableModules.js**

Immediately-Invoked Function Expressions (IIFE)

IIFE's can be used to avoid variable hoisting from within blocks, protect against polluting the global environment and simultaneously allow public access to methods while retaining privacy for variables defined within the function.

A common way to implement IIFE's is to enclose both the function expression and invocation in parentheses.

Example:
See **IIFE.js**

JavaScript Prototyping

Prototypes in JavaScript are kind of like instances of objects, and with each prototype you can add/remove/change elements which only affects the given prototype, it does not affect other prototypes of the same object, nor the object itself.

Example:
See **Objects.js**

User defined Callback Functions

Functions that takes a callback (another function) as an argument, are known as callback functions.

Example:
See **Callbacks.js**

Use the Debugger to explain about the basic "things" all objects inherits from object

I must admit that I'm unsure of what "all objects inherits from object", but I would guess that it has something to do with prototype.

Explain generally about node.js and NPM.

Node.js

"Node.js is a platform built on Chrome's JavaScript runtime for easily building fast, scalable network applications.
It uses an event-driven, non-blocking I/O model that makes it lightweight and efficient, perfect for data-intensive real-time applications that run across distributed devices
Keywords:

- Based on Googles V8 Engine.
- Event Driven.
- Highly targeted against async. Programming.
- Non-Blocking I/O.
- Easily Scalable".

Source: http://js2016.azurewebsites.net/node1/NodeIntro.html#2

NPM

NPM is a tool for version control and sharing code between developers. A developer can use the tool to share packages of code with others, so they can implement that into any project they're working on.

Provide examples of user defined reusable modules implemented in Node.js