JavaScript: Scope

What is Scope?

- Scope describes the accessibility of variables and functions in a program
- Some languages implicitly define scope (Javascript, Python)
- Other languages explicitly define scope using keywords or symbols (Ruby, Java, C#)

Here's an example of ruby setting scope explicitly

```
local_variable = 1 # visible in local scope (current method, class, etc)
@instance_variable = 2 # visible to instances of a class
$global_variable = 3 # visible anywhere in your Ruby script
```

Javascript's Implicit Scope

- JavaScript does not use special syntax or keywords to designate scope
- Scope depends on where your variable is defined
- In most cases, the level of indentation or curly braces shows it

```
// global scope - visible everywhere in this script
var word = "hello!";

// You can use word inside the function
function sayWord() {
  console.log(word);
}

sayWord(); // prints "hello!"
console.log(word); // also prints "hello!"
```

More Global Scope

```
var word = "hello!";

function sayWord() {
  console.log(word);
}

sayWord(); // prints "hello!"
word = "goodbye!";
sayWord(); // prints "goodbye!"
```

Local Scope

When something is locally scoped, by being contained inside of a function or a class, it won't be accessible outside of that function or class.

```
// Variable `word` is defined inside of this function
function sayWord() {
  var word = "beep";
  console.log(word);
}

sayWord(); // prints "beep"
console.log(word); // throws "ReferenceError: 'word' is not defined"
```

More Local Scope

```
function shoutify(word) {
  var newWord = word.toUpperCase() + "!";
  return newWord;
}

function whisperify(word) {
  var newWord = word.toLowerCase() + "...";
  return newWord;
}

var myWord = "Hello";
console.log(shoutify(myWord)); // Prints "HELLO!"
console.log(whisperify(myWord)); // Prints "hello..."
console.log(myWord); // Still prints "Hello"
console.log(newWord); // throws "ReferenceError: 'newWord' is not defined"
```

Scoped Functions

```
function outsideFunction() {

  function insideFunction() {
    console.log("hi!");
  }

  insideFunction();
}

outsideFunction(); // calls insideFunction, which then prints "hi!"
insideFunction(); // ReferenceError: insideFunction is not defined
```

Nested Scopes

```
var grapefruit = 1;

function outerFunction() {
  var apple = 2;

  function innerFunction() {
    var orange = 3;

    // At this level, we have access to variables 'grapefruit', 'apple', and 'orange'
    console.log(grapefruit + " " + apple + " " + orange); // prints "1 2 3", no errors
  }

  // At this level, we have access to variables 'grapefruit' and 'apple', but not 'orange'
}

// At this level, we have access only to variable 'grapefruit'
```

Gotchas To Avoid

"Bleeding" Scope

- Having variables in your scope also means you can alter them
- This is often undesirable in functions, if they alter global scope items

```
var word = "beep";

function wordDoubler() {
  word = word + word;
  return word;
}

console.log(word); // Prints "beep"
console.log(wordDoubler()); // Prints "beepbeep"
console.log(word); // Prints "beepbeep" also!
```

"Bleeding" Scope (Fixed)

```
var word = "beep";

function wordDoubler(wordToDouble) {
  return wordToDouble + wordToDouble;
}

console.log(word); // Prints "beep"
console.log(wordDoubler(word)); // Prints "beepbeep"
console.log(word); // Prints "beep" again
```

Same Name, Different Scope

- Variables can share the same name, but refer to different values
- This only works if they're defined in different scopes
- Collisions are handled by picking the most local scope

```
var number = 10;

function numberDoubler(number) {
  // Doubles the argument 'number', not the variable defined in the global scope
  return number * 2;
}

console.log(numberDoubler(number)); // Prints 20
console.log(numberDoubler(50)); // Prints 100, not 20
```

Same Name, Different Scope (pt2)

```
function greeting(shouldAskStatus) {
  var message = "Hello friend!";

  function askStatus() {
    // Return refers to the most local 'message', the one defined here.
    var message = "How's it going?";
    return message;
  }

  if (shouldAskStatus) {
    message = message + " " + askStatus();
  }

  return message;
}

console.log(greeting(false)); // Prints "Hello friend!"
console.log(greeting(true)); // Prints "Hello friend! How's it going?"
```