JavaScript

Part #2

Constants and variables - Scope

- const
  - Constant, local scope
- var
  - Variable, function/global scope
- let
  - Variable, local scope

const

- Defines constants

```
const number = 25;
```

Const (2)

- Try this in the browser and check the console:

```
const number = 25;
console.log(number);
```

Const (3)

- Try this in the browser and check the console:

```
console.log(number);
const number = 25;
```

Const (4)

- Try this in the browser and check the console:

```
const number = 25;
{
    const number = 30;
    console.log(number);
}
console.log(number);
```

Variables

- Try this in the browser and check the console:

```
var number = 25;
console.log(number);
```

Variables (2)

- Try this in the browser and check the console:

```
console.log(number);
var number = 25;
```

Variables (3)

- Try this in the browser and check the console:

```
console.log(number);
```

Variables (4)

- Try this in the browser and check the console:

```
let number = 25;
console.log(number);
```

Variables (5)

- Try this in the browser and check the console:

```
console.log(number);
let number = 25;
```

Scope

```
const number = 25;
{
    const number = 30;
    console.log(number);
}
console.log(number);
```

Scope

```
const number = 25;
{
    var number = 30;
    console.log(number);
}

console.log(number);
```

Scope

```
var number = 25;
{
    const number = 30;
    console.log(number);
}

console.log(number);
```

Scope

```
var number = 25;
{
    var number = 30;
    console.log(number);
}

console.log(number);
```

Scope

```
var number = 25;
{
    let number = 30;
    console.log(number);
}

console.log(number);
```

Scope

```
let number = 25;
{
    var number = 30;
    console.log(number);
}

console.log(number);
```

Question

```
var msg = "global";
{
    var msg = "in scope";
    alert(msg);
}
alert(msg);
```

- What are the outputs?

A. Global then in scope

B. in scope then in scope

C. Global then global

D. in scope then global

var Location Does Not Matter!!

- You can use a variable before it is declared. It will be undefined.
  - BUT you must declare this variable somewhere in the scope!
- But best practice is to declare all at the start of the function.

Rules of thumb

- Use **`const`** unless you need to change value
- Use **`let`** if you need to change the value BUT you don't need the variable to be function/global
- Use **`var`** if you need to change the value and you need function/global scope

function

```
function function_name([formal_parameters]) {
  -- body -
 }
```

- Return value is the parameter of `return`
  - If there is no `return`, or if the end of the function is reached, `undefined` is returned
  - If `return` has no parameter, `undefined` is returned
- `functions` are objects

```
refToFunction = fun;
    ...
refToFunction();  /* A call to fun */
```

- Functions are defined in the head of the HTML file

function

- No type checking, no number of parameters checking
- What happens to the parameters?

Params.js

```
 function params(a, b) {
        document.write("Function params was passed ",
            arguments.length, " parameter(s) <br />");
        document.write("Parameter values are: <br />");

        for (var arg = 0; arg < arguments.length; arg++)
          document.write(arguments[arg], "<br />");
        document.write("a="+a+" "+"b="+b+"<br />");
        document.write("<br />");
        }
// A test driver for function params
      params("Mozart");
      params("Mozart", "Beethoven");
      params("Mozart", "Beethoven", "Tchaikowsky");
```