

More! About! Functions!

Questions

- Where can you access argument values?
- What is a callback?
- What is an anonymous function?
- Where do callback arguments come from?

Scopes





© shop.lego

Scope: looking for values of variables

```
var name = 'Oakley';  
  
function greet() {  
  console.log('Hey, ' + name + '!');  
}  
  
greet();
```

Scope: looking for values of variables

```
var name = 'Oakley';  
  
function greet() {  
  console.log('Hey, ' + name + '!');  
}  
  
greet();
```

```
// Hey, Oakley!
```

Scope: looking for values of variables

```
var name = 'Oakley';

function greet() {
  var name = 'Milla';
  console.log('Hey, ' + name + '!');
}

greet();
```


Scope: looking for values of variables

```
var name = 'Oakley';  
  
function greet() {  
  var name = 'Milla';  
  console.log('Hey, ' + name + '!');  
}  
  
greet();
```

```
// Hey, Milla!
```

Function Scope

```
var name = 'Oakley';  
  
function greet() {  
  var name = 'Milla';  
  console.log('Hey, ' + name + '!');  
}  
  
greet();
```

Global Scope

```
function topLevel {  
  var name = 'Oakley';  
  
  function greet() {  
    var name = 'Milla';  
    console.log('Hey, ' + name + '!');  
  }  
  
  greet();  
}
```

Global Scope

```
function topLevel {  
  
  var name = 'Oakley';  
  
  function greet() {  
    var name = 'Milla';  
    console.log('Hey, ' + name + '!');  
  }  
  
  greet();  
}
```

PS: The `topLevel` function is imaginary

Scopes are often nested

```
function topLevel {  
  var name = 'Oakley';  
  
  function greet() {  
    var name = 'Milla';  
    console.log('Hey, ' + name + '!');  
  }  
  
  greet();  
}
```

Demo: examining scope in the dev tools

Hoisting

What on earth...

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```


What on earth...

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

```
// Hey undefined!
```

Variable declarations are "hoisted"

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

Variable declarations are "hoisted"

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

```
function greet() {  
  var name;  
  console.log('Hey, ' + name + '!');  
  name = 'Milla';  
}
```

Variable declarations are "hoisted"

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

```
function greet() {  
  var name;  
  console.log('Hey, ' + name + '!');  
  name = 'Milla';  
}
```

assignment is not!

That var in a for loop...

```
function sumNumbers(arr) {  
  var result = 0;  
  for (var i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```

That var in a for loop...

```
function sumNumbers(arr) {  
  var result = 0;  
  for (var i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```

```
function sumNumbers(arr) {  
  var result = 0;  
  var i;  
  for (i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```

Function expressions

Two ways to create

```
function sumNumbers(arr) {  
  var result = 0;  
  for (var i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```


Two ways to create

```
function sumNumbers(arr) {  
  var result = 0;  
  for (var i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```

```
var sumNumbers = function (arr) {  
  var result = 0;  
  for (var i=0; i < arr.length; i++) {  
    result = result + arr[i];  
  }  
  return result;  
}
```

Almost alike

```
function sumNumbers(arr) { }
```

```
var sumNumbers = function (arr) { };
```



Function hoisting

```
greet();  
  
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

Function hoisting

```
greet();  
  
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}
```

```
function greet() {  
  console.log('Hey, ' + name + '!');  
  var name = 'Milla';  
}  
  
greet();
```

Function expressions are *not* hoisted

```
greet();

var greet = function () {
  console.log('Hey, ' + name + '!');
  var name = 'Milla';
}
// womp womp
// TypeError: greet is not a function
```

Function expressions are *not* hoisted

```
greet();

var greet = function () {
  console.log('Hey, ' + name + '!');
  var name = 'Milla';
}
// womp womp
// TypeError: greet is not a function
```

```
var greet;

greet();

greet = function () {
  console.log('Hey, ' + name + '!');
  var name = 'Milla';
}
```

Array methods

forEach

```
var nums = [1, 2, 3, 4, 5];  
function log(what) {  
  console.log(what);  
}  
nums.forEach(log);
```


forEach with a function expression!

```
var nums = [1, 2, 3, 4, 5];  
function log(what) {  
  console.log(what);  
}  
nums.forEach(log);
```

forEach with a function expression!

```
var nums = [1, 2, 3, 4, 5];  
function log(what) {  
  console.log(what);  
}  
nums.forEach(log);
```

```
var nums = [1, 2, 3, 4, 5];  
nums.forEach(function (what) {  
  console.log(what);  
});
```

[MDN Docs](#)

All the arguments

```
var nums = [1, 2, 3, 4, 5];  
  
nums.forEach(function (what, i, arr) {  
  console.log(i + ": " + what + " is part of " + arr);  
});
```

Let's (square) dance

```
var nums = [1, 2, 3, 4, 5];  
var squares = [];  
  
function square(n) {  
    return n * n;  
}  
  
nums.forEach(function (num) {  
    squares.push(square(num));  
});
```

Array.map

```
var nums = [1, 2, 3, 4, 5];
var squares = [];

function square(n) {
  return n * n;
}

nums.forEach(function (num) {
  squares.push(square(num));
});
```

```
var nums = [1, 2, 3, 4, 5];

function square(n) {
  return n * n;
}

var squares = nums.map(square);
```

[MDN docs](#)

Filtering data

```
var nums = [1, 2, 3, -4, -5, 6, 7, 8, -9, -10];
var result = [];
for (var i=0; i < nums.length; i++) {
  if (nums[i] >= 0) {
    result.push(nums[i]);
  }
}
```

Filtering data

```
var nums = [1, 2, 3, -4, -5, 6, 7, 8, -9, -10];
var result = [];
for (var i=0; i < nums.length; i++) {
  if (nums[i] >= 0) {
    result.push(nums[i]);
  }
}
```

Or, with forEach

```
var nums = [1, 2, 3, -4, -5, 6, 7, 8, -9, -10];
var result = [];
nums.forEach(functions (num) {
  if (num >= 0) {
    result.push(num);
  }
});
```

Array.filter

```
var nums = [1, 2, 3, -4, -5, 6, 7, 8, -9, -10];
var result = [];
nums.forEach(functions (num) {
  if (num >= 0) {
    result.push(num);
  }
});
```

```
var nums = [1, 2, 3, -4, -5, 6, 7, 8, -9, -10];
function isPositive(num) {
  return num > 0;
}
var result = nums.filter(isPositive);
```

[MDN docs](#)

Questions

- Where can you access argument values?
- What is a callback?
- What is an anonymous function?
- Where do callback arguments come from?

finito.