Closure

Douglas Crockford

Block Scope

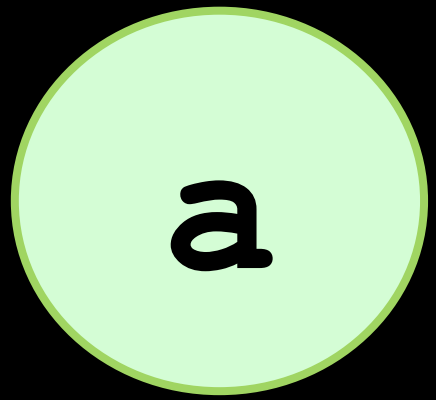
```
{
    let a;
    {
        let b;
        … a …
        … b …
    }
    … a …
}
```

Function Scope

```
function green() {
    let a;
    function yellow() {
        let b;
        … a …
        … b …
    }
    … a …
}
```

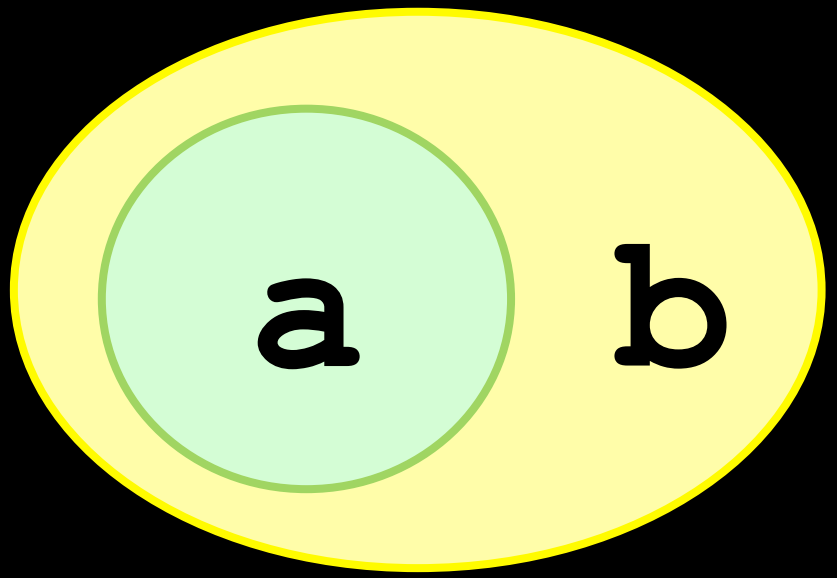
Function Scope

```
function green() {
    let a;
    function yellow() {
        let b;
        … a …
        … b …
    }
    … a …
}
```



Function Scope

```
function green() {
    let a;
    function yellow() {
        let b;
        … a …
        … b …
    }
    … a …
}
```



- Lisp [1958]
  - dynamic scope
  - nested functions
  - function values
- Algol 60 [1960]
  - lexical scope
  - nested functions
  - functions are not values
- C [1972]
  - lexical scope
  - functions cannot nest
  - functions are values

Inner survives the outer

```
function green() {
    let a;
    return function yellow() {
        let b;
        … a …
        … b …
    }
    … a …
}
```

Global

```
var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];

var digit_name = function (n) {
    return names[n];
};



alert(digit_name(3));    // 'three'
```

Slow

```
var digit_name = function (n) {
    var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];


    return names[n];
};



alert(digit_name(3));    // 'three'
```

Closure

```
var digit_name = (function () {
    var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];

    return function (n) {
        return names[n];
    };
}());


alert(digit_name(3));    // 'three'
```

Start Over

```
var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];

var digit_name = function (n) {
    return names[n];
};



alert(digit_name(3));    // 'three'
```

Immediate function returns a function

```
var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];
var digit_name = (function () {
    return function (n) {
        return names[n];
    };
}());


alert(digit_name(3));    // 'three'
```

Closure

```
var digit_name = (function () {
    var names = ['zero', 'one', 'two',
        'three', 'four', 'five', 'six',
        'seven', 'eight', 'nine'];

    return function (n) {
        return names[n];
    };
}());


alert(digit_name(3));    // 'three'
```

The Y Combinator

```
function y(le) {
    return (function (f) {
        return f(f);
    }(function (f) {
        return le(function (x) {
            return f(f)(x);
        });
    }));
}

var factorial = y(function (recur) {
    return function (n) {
        return n <= 2 ? n : n * recur(n - 1);
    };
});

var number120 = factorial(5);
```