Programming Fundamentals with JavaScript

Functions

Duke
UNIVERSITY

Functions: Complex Steps Not on Object

- Methods: invoked on objects
- Functions: *not* invoked on objects
  - Example: `print(x);`
  - Similar syntax, no "object."
- Can write your own!

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}


var y = square(4);
```

- Simple function: square a number

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

The keyword *function* says
"I'm about to define a new function"

- Simple function: square a number
  - Let us break down the syntax

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

The name of the function
(pretty much anything you want)

- Simple function: square a number
  - Let us break down the syntax

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

The parameter names, in parentheses
(separate multiple ones with commas)

- Simple function: square a number
  - Let us break down the syntax

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

Body is inside curly braces

The body of the function:
Statements that are what it does

- Simple function: square a number
  - Let us break down the syntax

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

Return statement:
return (expression) ;

- Simple function: square a number
  - Let us break down the syntax

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}


var y = square(4);
```

Function call: makes use of function

- Simple function: square a number
  - Let us break down the syntax

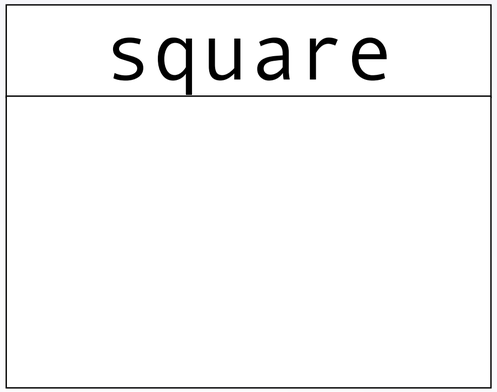
Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```

- Semantics:
  - Calling a function you defined

Writing Your Own Function

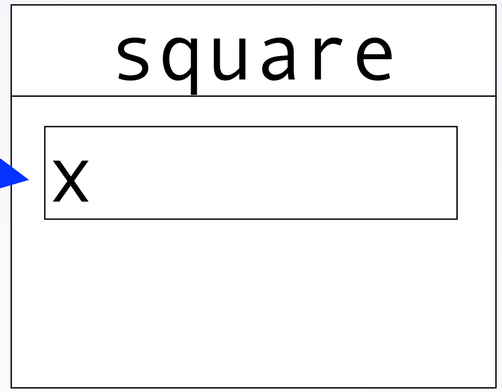
```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```



- Semantics:
  - Calling a function you defined

Writing Your Own Function

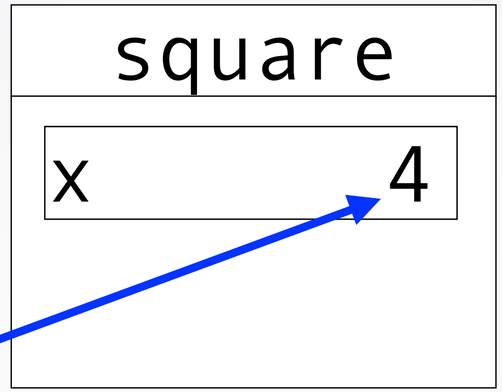
```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```



- Semantics:
  - Calling a function you defined

Writing Your Own Function

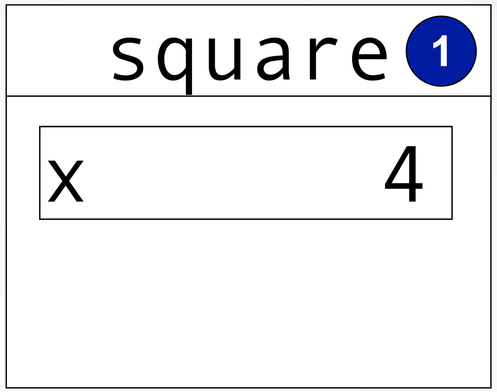
```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```



- Semantics:
  - Calling a function you defined

Writing Your Own Function

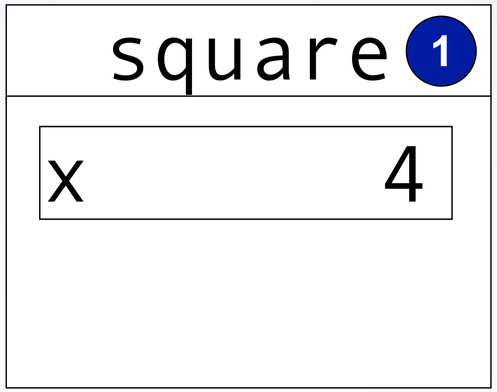
```
function square(x) {
    var ans = x * x;
    return ans;
}

var y = square(4);
```



- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}


var y = square(4);
```



- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}
```

square 1	
x	4
ans	16

```
var y = square(4);
```

- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(
    var ans = x *
    return ans;
}
```

My answer is 16

square 1	
x	4
ans	16

```
var y = square(4);
```

- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(
    var ans = x *
    return ans;
}
```

My answer is 16

square 1	
x	4
ans	16

```
var y = square(4);
```
1 **= 16**

- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}
```

```
var y = square(4);
```
= 16

- Semantics:
  - Calling a function you defined

Writing Your Own Function

```
function square(x) {
    var ans = x * x;
    return ans;
}


var y = square(4);
```

y	16

- Semantics:
  - Calling a function you defined

Why Methods and Functions?

- Why are methods and functions good?
  - Abstraction!
  - Separate interface from implementation
- Interface:
  - Call `image.getWidth()`, get with of image
- Implementation:
  - Code in DLTP library (not shown)
  - No need to see!