JavaScript – Complex Data Types

All high level languages (JavaScript, C++, Java, Python, etc.) have complex data types. We have already seen one in Javascript – arrays. And you have already used several in Python – lists and dictionaries.

There are two super important complex data types in JavaScript – objects and maps. In order to use the cool features of JavaScript, we need to understand both of these.

Objects

A JavaScript object has a number of subvariables, in an

unordered list.

in an ordered list, in contrast.

```
let book = {title: "The
Catcher in The Rye",
author: "J.D. Salinger",
Published: 1951,
keywords: ["classics",
"banned", "nostalgia"]
};
```

book.title ← string

book.author ← string

book.published ← int

book.keywords ← array of strings.

book.keywords[2] ← string

→ sub variables can be of different types

→ so, most like a Python list.

→ subvariables can be, in turn, other objects

```
let book = { title: "Hamlet",
             published: 1608,
             author: { firstName:
                       "William"
                       lastName:
                       "Shakespeare"
                     }
           };

Console.log( book.author.firstName );
```

↳ William

We can add functions to objects!
(called methods)

```
let book = { ——
```

```
    getAuthorName : function() {
        return this.author.firstName + " "
            + this.author.lastName;
    }
};
```

refer to "this" object, as opposed to some other object.

(Note: we can also define such functions outside of the original object definition)

A common technique is to have subvariables based on other subvariables. We use setter and getter methods to set and retrieve such subvariables, to maintain internal consistency, and to encapsulate the objects from the user.

Example :

```
let rectangle = {  width: 5,  height: 10,

    get area () {
        return this.width * this.height;
    },

    set area (value) {
        this.width = Math.sqrt
                    (2.0* value);
        this.height = this.width /2.0;
    }
};

:
:
:

let area = rectangle.area
// calls getter method !
console.log(area);
```

```
rectangle.area = 100;

// calls setter method!

Console.log(rectangle.width);
console.log(rectangle.width);
```