

Basics of Javascript Objects

Relevant Links

- Flanagan's book, sections 4.2, 4.4, 6.1.1, 6.2.1
- MDN's guide on objects¹
- MDN's reference on objects²

We will start with simple object literals for now.

Basics of Javascript Objects

- For the time being, we will be working with objects as simply key-value pairs. So think of them more like Python's *dictionaries*.
- An object is a dynamic collection of key-value pairs. The keys are usually called *properties*.
- Almost everything in Javascript is an object.
- Object literals are enclosed in curly braces:

```
var a = {  
  foo: 123,  
  "bar": "hello",  
  "properties can be any string": "values can be anything",  
  even: { other: "objects" }  
};
```

The keys can be written without quotes around them if there is no ambiguity in doing so.

- Accessing a property: `a["foo"]`, `a.foo`.
- You can also access a property if you have it as a variable value:

```
var b = "foo";  
a[b];           // same as a["foo"]
```
- Setting a property: `a["foo"] = 3`, `a.foo = 3`.
- You may delete properties, though this is rare and to be avoided: `delete a.foo`.
- You can chain accesses: `a.even.other`.
- You may use `hasOwnProperty` to determine if an object has a specific property:
`a.hasOwnProperty("bar")`

¹https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide/Working_with_Objects

²https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Object

- Two special values of importance: `null`, `undefined`. Both tend to indicate the absence of a value. The difference is that the former of those is an object:

```
typeof null;  
typeof undefined;
```

- If you try to access a non-existent property, the result is `undefined`.
- You can also set the value of a property to equal `undefined`. This is different from not having that property.