Retrieving Property Names with `Object.getOwnPropertyNames` and `Object.keys`

Originally published in the A Drip of JavaScript newsletter.

In past versions of JavaScript it was fairly painful to figure out what properties an object possessed. Essentially you would need to manually iterate over the object and filter out inherited properties, like so:

```
var charactersBooks = {
    Frodo: "Lord of the Rings",
    Aslan: "Chronicles of Narnia",
};

var characters = [ ];

for (var prop in charactersBooks) {
    if (charactersBooks.hasOwnProperty(prop)) {
        characters.push(prop);
    }
}

// Outputs: ["Frodo", "Aslan"]
console.log(characters);
```

But with ECMAScript 5 we get access to `Object.keys` which eliminates this tedious boilerplate.

```
var charactersBooks = {
    Frodo: "Lord of the Rings",
    Aslan: "Chronicles of Narnia",
};

var characters = Object.keys(charactersBooks);

// Outputs: ["Frodo", "Aslan"]
console.log(characters);
```

As might be expected, `Object.keys` only retrieves the names of properties that are declared directly on the object. It doesn't get the names of inherited properties. In addition, it only retrieves the names of enumerable properties. Non-enumerable property names are omitted.

But ES5 also gave us another method called `Object.getOwnPropertyNames` which is less strict about which property names it will retrieve. Like `keys`, `getOwnPropertyNames` will only retrieve "own" properties. However, it will retrieve the names of non-enumerable properties.

Not sure what enumerable means in this context? Non-enumerable properties are essentially properties that shouldn't be "counted" or iterated over for some reason. (More discussion to come in future drips.) The simplest example is probably an array's `length` property. Let's see how our functions treat it.

```
var authors = ["Tolkien", "Lewis"];

// Outputs: ["0", "1"]
console.log(Object.keys(authors));

// Outputs: ["0", "1", "length"]
console.log(Object.getOwnPropertyNames(authors));
```

As you can see, `Object.keys` skipped over the non-enumerable `length` property while `Object.getOwnPropertyNames` did not.

How do you know when to use one or the other? My personal rule is to always default to using `Object.keys` unless I specifically need the name of non-enumerable properties.

Because both of these functions are part of the ES5 spec, they are not available in older browsers like IE8. And unfortunately, it is impossible to backport `Object.getOwnPropertyNames`. However, `Object.keys` can be backported with something like the ES5 Shim library.
