Move DOM attributes to prototype chains
What may happen and how to fix

Author: Yuki Shiino <yukishiino@chromium.org>

Typical issues

The followings are typical issues that web developers would face to.

Issue 1: JSON.stringify(domObject) returns {}
or some properties are missing in the returned object

`JSON.stringify` stringifies only own properties, and does not traverse prototype chains. Before the change, we had implemented DOM attributes as own properties, so `JSON.stringify` worked for DOM attributes, but it was incorrect. The correct behavior is that DOM attributes are ignored by `JSON.stringify` unless DOM object has an own serializer.

How to fix?

Create a temporary object (regular object, not a DOM object) which stores the values of DOM attributes you'd like to stringify.

```
var tmp = {attr1: domObject.attr1, attr2: domObject.attr2};
var s = JSON.stringify(tmp);
```

More generally, you can copy all the DOM attributes in a DOM object into an Object and then stringify it.

```
function stringifyDOMObject(object)
{
    function deepCopy(src) {
        if (typeof src != "object")
            return src;
        var dst = Array.isArray(src) ? [] : {};
        for (var property in src) {
            dst[property] = deepCopy(src[property]);
        }
        return dst;
    }
    return JSON.stringify(deepCopy(object));
}
var s = stringifyDOMObject(domObject);
```

For those who are working on specs, consider to support `stringifier` so that `JSON.stringify` work for a DOM object.

Issue 2: Assignment to readonly attributes throws an exception in strict mode

In strict mode, assignment to readonly DOM attributes throws a `TypeError` exception, as the Web IDL spec requires. After the change, Blink detects this error more correctly and accurately.

Example

```
function foo() {
  'use strict';
  document.contentType = 'foo';
}
foo();
```

where `document.contentType` is a readonly attribute, so a `TypeError` will be thrown. When run on console, an error message looks like:

```
Uncaught TypeError: Cannot set property contentType of #<Document>
which has only a getter
    at foo (<anonymous>:2:53)
    at <anonymous>:2:1
    at Object.InjectedScript._evaluateOn (<anonymous>:895:140)
    at Object.InjectedScript._evaluateAndWrap (<anonymous>:828:34)
    at Object.InjectedScript.evaluate (<anonymous>:694:21)
```

How to fix?

1. Remove such code that assigns to readonly DOM attributes.
2. Surround such code with try-catch block and handle the error appropriately.
3. Stop using strict mode (not recommended). Then, assignment to readonly attributes is simply ignored. The value doesn't change as same as before.

Issue 3: Cannot detect a feature that must be existing by using hasOwnProperty.

Since DOM attributes were moved to prototype chains, the attributes are no longer own properties. So `domObject.hasOwnProperty('attribute')` returns `false`. If you're trying to detect a feature by using `hasOwnProperty`, it fails and seems as if the feature were not supported.

Example

```
if (!domObject.hasOwnProperty('maybeSupportedFeature')) {
  // A feature is not supported.
}
```

How to fix?

Use `in` operator instead.

```
if (!('maybeSupportedFeature' in domObject)) {
  // A feature is not supported.
}
```