**Disadvantage**

1. Other developers using your code will probably expect the built-in JavaScript methods to work consistently and will not expect your additions.
2. Properties you add to the prototype may show up in loops that don't use hasOwnProperty(), so they can create confusion.

**Augment build-in prototypes under all of the conditions below**:

1. It's expected that future ECMAScript versions or JavaScript implementations will implement this functionality as a built-in method consistently. For example, you can add methods described in ECMAScript 5 while waiting for the browsers to catch up. In this case you're just defining the useful methods ahead of time.
2. You check if your custom property or method doesn't exist already—maybe already implemented somewhere else in the code or already part of the JavaScript engine of one of the browsers you support.
3. You clearly document and communicate the change with the team.

If these three conditions are met, you can proceed with the custom addition to the prototype, following this pattern:

if (typeof Object.protoype.myMethod !== "function") {

Object.protoype.myMethod = function () {

// implementation...

};

}