# Chapter 66: Detecting browser

Browsers, as they have evolved, offered more features to JavaScript. But often these features are not available in all browsers. Sometimes they may be available in one browser, but yet to be released on other browsers. Other times, these features are implemented differently by different browsers. Browser detection becomes important to ensure that the application you develop runs smoothly across different browsers and devices.

## Section 66.1: Feature Detection Method

This method looks for the existence of browser specific things. This would be more difficult to spoof, but is not guaranteed to be future proof.

```
// Opera 8.0+
var isOpera = (!!window.opr && !!opr.addons) || !!window.opera || navigator.userAgent.indexOf('
OPR/') >= 0;

// Firefox 1.0+
var isFirefox = typeof InstallTrigger !== 'undefined';

// At least Safari 3+: "[object HTMLElementConstructor]"
var isSafari = Object.prototype.toString.call(window.HTMLElement).indexOf('Constructor') > 0;

// Internet Explorer 6-11
var isIE = /*@cc_on!@*/false || !!document.documentMode;

// Edge 20+
var isEdge = !isIE && !!window.StyleMedia;

// Chrome 1+
var isChrome = !!window.chrome && !!window.chrome.webstore;

// Blink engine detection
var isBlink = (isChrome || isOpera) && !!window.CSS;
```

Successfully tested in:

- Firefox 0.8 - 44
- Chrome 1.0 - 48
- Opera 8.0 - 34
- Safari 3.0 - 9.0.3
- IE 6 - 11
- Edge - 20-25

Credit to Rob W

## Section 66.2: User Agent Detection

This method gets the user agent and parses it to find the browser. The browser name and version are extracted from the user agent through a regex. Based on these two, the `<browser name> <version>` is returned.

The four conditional blocks following the user agent matching code are meant to account for differences in the user agents of different browsers. For example, in case of opera, since it uses Chrome rendering engine, there is an additional step of ignoring that part.

Note that this method can be easily spoofed by a user.

```
navigator.sayswho= (function(){
    var ua= navigator.userAgent, tem,
    M= ua.match(/(opera|chrome|safari|firefox|msie|trident(?=\/))\/?\s*(\d+)/i) || [];
    if(/trident/i.test(M[1])){
        tem=  /\brv[ :]+(\d+)/g.exec(ua) || [];
        return 'IE '+(tem[1] || '');
    }
    if(M[1]=== 'Chrome'){
        tem= ua.match(/\b(OPR|Edge)\/(\d+)/);
        if(tem!= null) return tem.slice(1).join(' ').replace('OPR', 'Opera');
    }
    M= M[2]? [M[1], M[2]]: [navigator.appName, navigator.appVersion, '-?'];
    if((tem= ua.match(/version\/(\d+)/i))!= null) M.splice(1, 1, tem[1]);
    return M.join(' ');
})();
```

Credit to kennebec

## Section 66.3: Library Method

An easier approach for some would be to use an existing JavaScript library. This is because it can be tricky to guarantee browser detection is correct, so it can make sense to use a working solution if one is available.

One popular browser-detection library is Bowser.

Usage example:

```
if (bowser.msie && bowser.version >= 6) {
    alert('IE version 6 or newer');
}
else if (bowser.firefox) {
    alert('Firefox');
}
else if (bowser.chrome) {
    alert('Chrome');
}
else if (bowser.safari) {
    alert('Safari');
}
else if (bowser.iphone || bowser.android) {
    alert('iPhone or Android');
}
```