Writing tree-shakeable code

Tree shaking has become one of the most popular performance features when working with bundlers.

The idea behind tree shanking is that:

1. You declare all of your imports and exports for each of your modules
2. Your bundler (Webpack, Rollup, and so on) analyzes your dependency tree during the compilation step
3. Any code that appears not to be used is dropped from the final bundle, or 'tree-shaken'.

That means that our library code should export individual functions rather than classes. It would be tempting to do something like this:

```
export class Number {
  constructor(num) {
    this.num = num;
  }
  add(otherNum) {
    return this.num + otherNum;
  }
  substract(otherNum) {
    return this.num - otherNum;
  }
}
```

While this is perfectly fine, it will prevent bundlers from tree shaking the code since we're exporting the entire class and all its methods regardless of whether they are used or not, the bundler has no way of knowing.

Instead, export functions individually. This way, the bundler will know that if the function is not used then it can be tree shaken out of the final bundle.

```
export function add(num, otherNum) {
  return num + otherNum
}

export function substract(num, otherNum) {
  return num - otherNum;
}
```

Keep your module functions side-effect free. Unless the bundler can be 100% certain that a function will not be used in your code, it will err on the side of caution and not tree shake the affected function.

Specifics of how to make sure tree shaking works depend on the tool chain that you use. Daniel Brain's JavaScript tree shaking, like a pro provides a good introduction to tree shaking with WebPack and Tree-Shaking in JavaScript with Rollup gives you a good starting point for tree shaking your code using Rollup.