**Lab 2 - Modern Web Browsers & JS Engine**

**How JS engine optimizes JavaScript code?**

Turbofan’s optimizations improve the net performance of JavaScript by mitigating the impact of bad JavaScript. Nevertheless, understanding these optimizations can provide further speedups.

To make it run faster, the bytecode can be sent to the optimizing compiler along with profiling data. The optimizing compiler makes certain assumptions based on the profiling data it has, and then produces highly optimized machine code.

If at some point one of the assumptions turns out to be incorrect, the optimizing compiler deoptimizes and goes back to the interpreter.

Here are some ideas for improving performance by leveraging optimizations in V8:

* Changing object properties results in new hidden classes.
* Changing order of object properties results in new hidden classes, as ordering is included in object shape.
* Functions change object shape based on the value type at a specific argument position. If this type changes, the function is deoptimized and re-optimized.
* Do not define classes in the function scope.
* Crankshaft formerly used byte count of a function to determine whether to inline a function.
* Try blocks were formerly prone to costly optimization-deoptimization cycles.