LUASCRIPT Development TODO

Week 5 - Phase 1A-1D Critical Stabilization

Phase 1A: Runtime Compatibility Fixes

Priority: Critical
Target: Complete by Week 5 Day 3

- [ ] **Memory Management**
- [ ] Fix memory leaks in string concatenation
- [ ] Optimize garbage collection triggers
- [ ] Resolve stack overflow in deep recursion
- [ ] **Cross-Platform Compatibility**
- [ ] Test and fix Windows execution issues
- [ ] Resolve macOS path handling problems
- [ ] Standardize line ending handling
- [ ] **Runtime Stability**
- [ ] Fix segmentation faults in edge cases
- [ ] Improve error recovery mechanisms
- [ ] Stabilize variable scope resolution

Phase 1B: Documentation Accuracy

Priority: High
Target: Complete by Week 5 Day 5

- [x] **Core Documentation Updates**
- [x] Update README.md with accurate status
- [x] Create comprehensive TODO.md
- [x] Establish PROJECT_STATUS.md
- [ ] **API Documentation**
- [ ] Document all implemented functions
- [ ] Create usage examples for each feature
- [ ] Add parameter and return value specifications
- [ ] **User Guide**
- [ ] Write installation instructions
- [ ] Create getting started tutorial
- [ ] Document language syntax and features

Phase 1C: Testing Stabilization

Priority: High
Target: Complete by Week 5 Day 7

- [ ] **Test Suite Reliability**
- [ ] Fix flaky tests in parser module
- [ ] Stabilize performance benchmarks
- [ ] Improve test isolation
- [ ] **Coverage Expansion**
- [ ] Add edge case tests for all operators
- [ ] Test error handling paths
- [ ] Add integration tests
- [ ] **Automated Testing**
- [ ] Set up CI/CD pipeline
- [ ] Configure automated test runs
- [ ] Add performance regression detection

Phase 1D: Performance Validation

Priority: Medium
Target: Complete by Week 5 End

- [ ] **Benchmark Verification**
- [ ] Validate claimed performance improvements
- [ ] Document actual vs. expected performance
- [ ] Identify performance bottlenecks
- [ ] **Optimization Validation**
- [ ] Verify optimization effectiveness
- [ ] Profile memory usage patterns
- [ ] Test performance under load

Week 6+ Future Tasks

Core Language Features

- [ ] Implement table/array data structures
- [ ] Add object-oriented programming support
- [ ] Implement module system
- [ ] Add coroutine support

Advanced Features

- [ ] Implement async/await functionality
- [ ] Add pattern matching
- [ ] Implement type inference system
- [ ] Add debugging support

Tooling and Ecosystem

- [ ] Create language server protocol support
- [ ] Build package manager
- [ ] Develop IDE plugins
- [ ] Create online playground

Performance and Optimization

- [ ] Implement JIT compilation
- [ ] Add bytecode generation
- [ ] Optimize standard library
- [ ] Implement parallel execution

Completed Tasks (Weeks 1-4)

Week 1: Foundation

- [x] Project structure setup
- [x] Basic lexer implementation
- [x] Initial parser framework
- [x] Git repository initialization

Week 2: Core Parser

- [x] Expression parsing
- [x] Statement parsing
- [x] AST node definitions
- [x] Syntax error handling

Week 3: Interpreter Engine

- [x] AST evaluation engine
- [x] Variable environment
- [x] Function call mechanism
- [x] Control flow execution

Week 4: Testing and Documentation

- [x] Test framework setup
- [x] Basic test suite
- [x] Performance benchmarking
- [x] Initial documentation

Notes

- **Critical Path**: Phase 1A must be completed before advancing to Phase 1B
- **Dependencies**: Some Phase 1C tasks depend on Phase 1A completion
- **Testing**: All fixes must include corresponding tests
- **Documentation**: All changes must be documented

Review Schedule

- **Daily**: Progress review and task updates
- **Weekly**: Phase completion assessment
- **Milestone**: Week 5 completion review before Week 6 planning

---

Last Updated: Week 5 Start
Next Review: Daily standup