```
classDiagram
    %% Core Data Structures
    class State {
        -id: string
        -value: any
        -transitions: Map~string, State~
        -metadata: StateMetadata
        +getId(): string
        +getValue(): any
        +getTransitions(): Map
        +getMetadata(): StateMetadata
        +addTransition(symbol: string, target: State): void
        +removeTransition(symbol: string): boolean
        +getNextState(symbol: string): State
        +hasTransition(symbol: string): boolean
        +computeStateSignature(classes: Map): string
        +isEquivalentTo(other: State, alphabet: Set): boolean
        +setEquivalenceClass(classId: number): void
        +clone(): State
    }

    class StateMachine {
        -states: Map~string, State~
        -initialState: State
        -currentState: State
        -alphabet: Set~string~
        -isMinimized: boolean
        +addState(id: string, value: any): State
        +getState(id: string): State
        +setInitialState(stateId: string): void
        +addTransition(fromId: string, symbol: string, toId: string): void
        +transition(symbol: string): State
        +reset(): void
        +processSequence(symbols: string[]): State
        +accepts(symbols: string[]): boolean
        +getReachableStates(): Set~State~
        +removeUnreachableStates(): number
    }

    class StateMachineMinimizer {
        -options: MinimizationOptions
        +minimize(stateMachine: StateMachine): StateMachine
        -computeEquivalenceClasses(stateMachine: StateMachine): Map
        -createMinimizedMachine(original: StateMachine, classes: Map):
StateMachine
        -applyMemoryOptimizations(stateMachine: StateMachine): void
        -collectMetrics(original: StateMachine, minimized: StateMachine):
MinimizationMetrics
    }

    class EquivalenceClassComputer {
        +computeEquivalenceClasses(stateMachine: StateMachine): Map
```

```
        -splitPartition(partition: Set, allPartitions: Set, alphabet: Set): Map
        -computeTransitionSignature(state: State, partitions: Set, alphabet: Set):
string
        -findPartitionIndex(state: State, partitions: Set): number
        +findEquivalenceClass(state: State, classes: Map): number
        +areStatesEquivalent(state1: State, state2: State, classes: Map): boolean
        +computeStateSignatures(states: State[], classes: Map): Map
    }

    %% Caching System
    class StateMachineCache {
        -cache: Map~string, CachedStateMachine~
        -options: CacheOptions
        -metrics: CacheMetrics
        +get(key: string): StateMachine
        +set(key: string, machine: StateMachine): void
        +has(key: string): boolean
        +delete(key: string): void
        +clear(): void
        +getMetrics(): CacheMetrics
        -generateKey(machine: StateMachine): string
        -cleanup(): void
    }

    class CacheableStateMachine {
        -machine: StateMachine
        -cache: StateMachineCache
        +transition(symbol: string): State
        +processSequence(symbols: string[]): State
        +serialize(): string
        +deserialize(data: string): StateMachine
        -getCachedTransition(symbol: string): State
        -setCachedTransition(symbol: string, target: State): void
    }

    %% DOP Adapter Core
    class DOPAdapter {
        -dataModel: DataModel
        -behaviorModel: BehaviorModel
        -stateMachineMinimizer: StateMachineMinimizer
        -stateCache: StateMachineCache
        +createFromFunctional(config: FunctionalConfig): Component
        +createFromClass(componentClass: Class): Component
        +getState(): State
        +setState(newState: State): void
        +applyTransition(transitionName: string, payload: any): void
        +registerTransition(name: string, transitionFn: Function): void
        +optimizeStateMachine(): void
        +precomputeTransition(name: string, statePattern: Object): void
    }

    class DataModel {
        -immutableState: Object
        -transitionMaps: Map~string, Function~
```

```
        -validationRules: Map~string, Function~
        -equivalenceClasses: Map~number, Set~State~~
        -optimizedAST: ASTNode
        +getState(): Object
        +setState(newState: Object): void
        +getTransitionMap(name: string): Function
        +setTransitionMap(name: string, fn: Function): void
        +validateState(state: Object): boolean
        +validateTransition(name: string, state: Object, payload: any): boolean
        +computeEquivalenceClasses(): Map
        +optimizeAST(): ASTNode
    }

    class BehaviorModel {
        -stateTransitions: Map~string, Function~
        -minimizationLogic: StateMachineMinimizer
        -eventHandlers: Map~string, Function~
        -lifecycleHooks: Map~string, Function~
        -diffingAlgorithm: Function
        +applyTransition(name: string, state: Object, payload: any): Object
        +handleEvent(name: string, payload: any): void
        +registerLifecycleHook(name: string, handler: Function): void
        +triggerLifecycleHook(name: string, args: any[]): void
        +computeDiff(oldState: Object, newState: Object): Array
        +applyDiff(target: Object, diff: Array): Object
    }

    %% API Interfaces
    class Component {
        <<interface>>
        +state: Object
        +trigger(event: string, payload?: any): void
        +subscribe(listener: Function): Function
    }

    class FunctionalComponent {
        +initialState: Object
        +transitions: Object
        +render(state: Object, trigger: Function): RenderOutput
        +state: Object
        +trigger(event: string, payload?: any): void
        +subscribe(listener: Function): Function
    }

    class OOPComponent {
        +initialState: Object
        +[methodName: string]: Function
        +render(state: Object): RenderOutput
        +state: Object
        +trigger(event: string, payload?: any): void
        +subscribe(listener: Function): Function
        +_onMount(): void
        +_onUpdate(prevState: Object, newState: Object): void
        +_onUnmount(): void
```

```
}

%% AST & Parsing
class ASTNode {
    -type: string
    -value: any
    -children: ASTNode[]
    -parent: ASTNode
    -metadata: NodeMetadata
    +addChild(node: ASTNode): void
    +removeChild(node: ASTNode): void
    +clone(): ASTNode
    +computeSignature(): string
}

class HTMLTokenizer {
    -input: string
    -position: number
    -tokens: Token[]
    +tokenize(): Token[]
    -consumeToken(): Token
    -readTag(): Token
    -readAttribute(): Attribute
    -readText(): Token
    -readComment(): Token
}

class HTMLParser {
    -tokenizer: HTMLTokenizer
    -ast: ASTNode
    -states: Set~State~
    -equivalenceClasses: Map
    +parse(input: string): ASTNode
    -processToken(token: Token): void
    -buildAST(tokens: Token[]): ASTNode
    -minimizeStates(): void
    -optimizeAST(ast: ASTNode): ASTNode
}

class HTMLAstOptimizer {
    -stateClasses: Map
    -nodeSignatures: Map
    -minimizedNodes: WeakMap
    +optimize(ast: ASTNode): ASTNode
    -buildStateClasses(ast: ASTNode): void
    -computeNodeSignature(node: ASTNode): string
    -optimizeNode(node: ASTNode): ASTNode
    -optimizeChildren(children: ASTNode[]): ASTNode[]
    -mergeAdjacentNodes(children: ASTNode[]): ASTNode[]
    -applyMemoryOptimizations(node: ASTNode): void
    -computeOptimizationMetrics(original: ASTNode, optimized: ASTNode): Object
}

%% Diff Engine
```

```
    class DiffPatchEngine {
        -options: DiffOptions
        +diff(oldTree: ASTNode, newTree: ASTNode): Patch[]
        +patch(target: ASTNode, patches: Patch[]): ASTNode
        -findNodeDifferences(oldNode: ASTNode, newNode: ASTNode): Difference[]
        -createPatch(differences: Difference[]): Patch[]
        -applyPatch(target: ASTNode, patch: Patch): void
        -optimizePatch(patches: Patch[]): Patch[]
    }

    %% Relationships
    DOPAdapter ..> StateMachineMinimizer : uses
    DOPAdapter "1" *-- "1" DataModel : contains
    DOPAdapter "1" *-- "1" BehaviorModel : contains
    DOPAdapter ..> StateMachineCache : uses
    DOPAdapter ..> FunctionalComponent : creates
    DOPAdapter ..> OOPComponent : creates

    FunctionalComponent --|> Component : implements
    OOPComponent --|> Component : implements

    StateMachineMinimizer ..> EquivalenceClassComputer : uses
    StateMachineMinimizer ..> StateMachine : optimizes

    State --o StateMachine : composed in

    CacheableStateMachine --* StateMachine : wraps
    CacheableStateMachine ..> StateMachineCache : uses

    HTMLParser ..> HTMLTokenizer : uses
    HTMLParser ..> HTMLAstOptimizer : uses
    HTMLParser --* ASTNode : creates

    HTMLAstOptimizer ..> ASTNode : optimizes

    DiffPatchEngine ..> ASTNode : compares and modifies

    BehaviorModel ..> DiffPatchEngine : uses
    DataModel ..> HTMLAstOptimizer : uses
    ```
```
```