

"Deep Path Properties" in Record literals

For Stage 1 @ TC39 June 2020

Rick Button & Robin Ricard
Bloomberg

*Advisor: Daniel Ehrenberg
Igalia*

// mutable objects

```
let simple = { foo: "foo", bar: "bar" };  
simple.bar = "baz";
```

```
let complex = {  
  foo: {  
    arr: [{ counter: 0 }, { counter: -1 }]  
  }  
};  
complex.foo.arr[0].counter = 1;
```

// with Immer

```
let complex2 = Immer.produce(complex, draft => {  
  draft.foo.arr[0].counter = 1;  
});
```

// with Immutable.js

```
let immutableComplex = Immutable.fromJS(complex);  
let complex2 = immutableComplex.merge({  
  foo: {  
    arr: immutableComplex.get("arr").zipWith(  
      (a, b) => a.merge(b),  
      [  
        { counter: 1 },  
        {},  
      ],  
    ),  
  },  
});
```

// copy records, but only shallow

```
let simple = #{ foo: "foo", bar: "bar" };
```

```
let simple2 = #{ ...simple, bar: "baz" };
```

```
let complex = #{  
  foo: {  
    arr: #{ #{ counter: 0 }, #{ counter: -1 } },  
  },  
};
```

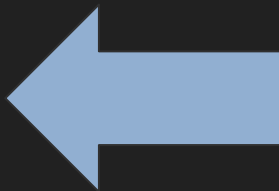
```
let complex2 = #{ ...complex, ??? };
```

// verbose, error-prone

```
let complex2 = #{  
  ...complex,  
  foo: #{  
    ...complex.foo,  
    arr: #[ #{  
      ...complex.foo.arr[0],  
      counter: 1,  
    }, ...complex.foo.arr.slice(1)],  
  },  
};
```

```
// new proposal
```

```
let complex = #{  
  foo: {  
    arr: [# { counter: 0 }, # { counter: -1 }],  
  },  
};  
  
let complex2 = #{  
  ...complex,  
  foo.arr[0].counter: 1,  
};
```



```
let state1 = #{  
  counters: #[  
    #{ name: "Counter 1", value: 1 },  
    #{ name: "Counter 2", value: 0 },  
    #{ name: "Counter 3", value: 123 },  
  ],  
  metadata: #{  
    lastUpdate: 1584382969000,  
  },  
};
```



```
let state1 = #{  
  counters: #[  
    #{ name: "Counter 1", value: 1 },  
    #{ name: "Counter 2", value: 0 },  
    #{ name: "Counter 3", value: 123 },  
  ],  
  metadata: #{  
    lastUpdate: 1584382969000,  
  },  
};
```

```
let state2 = #{  
  ...state1,  
  counters: #[  
    #{  
      ...state1.counters[0],  
      value: 2,  
    },  
    #{  
      ...state1.counters[1],  
      value: 1,  
    },  
    ...state1.counters,  
  ],  
  metadata: #{  
    ...state1.metadata,  
    lastUpdate: 1584383011300,  
  },  
}
```

```
let state1 = #{  
  counters: #[  
    #{ name: "Counter 1", value: 1 },  
    #{ name: "Counter 2", value: 0 },  
    #{ name: "Counter 3", value: 123 },  
  ],  
  metadata: #{  
    lastUpdate: 1584382969000,  
  },  
};
```

```
let state2 = #{  
  ...state1,  
  counters[0].value: 2,  
  counters[1].value: 1,  
  metadata.lastUpdate: 1584383011300,  
};
```

What happens if the deep path doesn't already exist?

```
const one = #{ a: #{ } };  
#{ ...one, a.b.c: "foo" }; // throws TypeError
```

```
#{ ...one, a.b[0]: "foo" }; // throws TypeError
```

// both seem like reasonable results, hence ambiguity

```
#{ a: #{ b: #[123] } }  
#{ a: #{ b: #{ 0: 123 } } }
```

What happens a deep path property attempts to set a non-number-like key on a Tuple?

```
const one = #{ a: #[1,2,3] };  
#{ ...one, a.foo: 4 }; // throws TypeError
```

Tuples cannot have non-number-like keys, therefore invalid to create one via deep paths.

Open Question: What about objects?

Deep path properties would be useful for object creation, but semantics are harder to understand than for Record literals.

Requires more investigation to discover if reasonable.

Stage 1?

Discuss!