Let's create some JavaScript assignments to practice enhanced object literals. These will cover shorthand property names, shorthand method names, computed property names, and the \_\_proto\_\_ property (though its direct use is generally discouraged in modern JavaScript).

Shorthand Property Names:

1. Creating a Person Object:

JavaScript

const name = "Alice";

const age = 30;

const city = "New York";

// Use enhanced object literal shorthand to create a person object with these properties.

1. Product Object:

JavaScript

const productName = "Laptop";

const productPrice = 999;

// Create a product object using shorthand property names.

Shorthand Method Names:

1. Greeting Function in Object:

JavaScript

const person = {

name: "Bob",

greet: function() {

console.log("Hello, " + this.name);

}

};

// Rewrite the 'greet' method using the shorthand method name syntax.

1. Calculator Object:

JavaScript

const calculator = {

add: function(a, b) {

return a + b;

},

subtract: function(a, b) {

return a - b;

}

};

// Rewrite the 'add' and 'subtract' methods using shorthand method names.

Computed Property Names:

1. Dynamic Property Keys:

JavaScript

const propertyName = "city";

const person = {

name: "Alice",

[propertyName]: "London" // Use a computed property name

};

// What is the value of person.city?

1. Property from Variable:

JavaScript

const key = "age";

const value = 25;

// Create an object with a property whose name is the value of 'key' and whose value is the value of 'value'.

\_\_proto\_\_ (Less Common Now):

*Note: Direct manipulation of \_\_proto\_\_ is generally discouraged now. Modern JavaScript uses Object.setPrototypeOf() for setting prototypes.1 These examples are for understanding the concept, but avoid direct \_\_proto\_\_ usage in production code.*

1. Setting Prototype (Discouraged):

JavaScript

const animal = {

eats: true

};

const dog = {

name: "Buddy",

\_\_proto\_\_: animal // Set the prototype of 'dog' to 'animal' (avoid this in modern code)

};

// How can you access the 'eats' property of 'dog'?

1. Object Creation with Prototype (Discouraged):

JavaScript

const parent = {

value: 10

};

// Create an object 'child' whose prototype is 'parent' using the now less common approach.

Combining Features:

1. Complex Object Literal:

JavaScript

const name = "Charlie";

const age = 35;

const city = "San Francisco";

const greet = function() {

console.log("Hi, I'm " + this.name);

};

const propertyName = "occupation";

const propertyValue = "Software Engineer";

// Create an object using all the enhanced object literal features: shorthand properties, shorthand method, computed property.

These assignments cover the key aspects of enhanced object literals. They should give you a good understanding of how to use these features to write more concise and expressive JavaScript code. Remember to experiment and try different combinations to solidify your understanding. Let me know if you have any other questions.