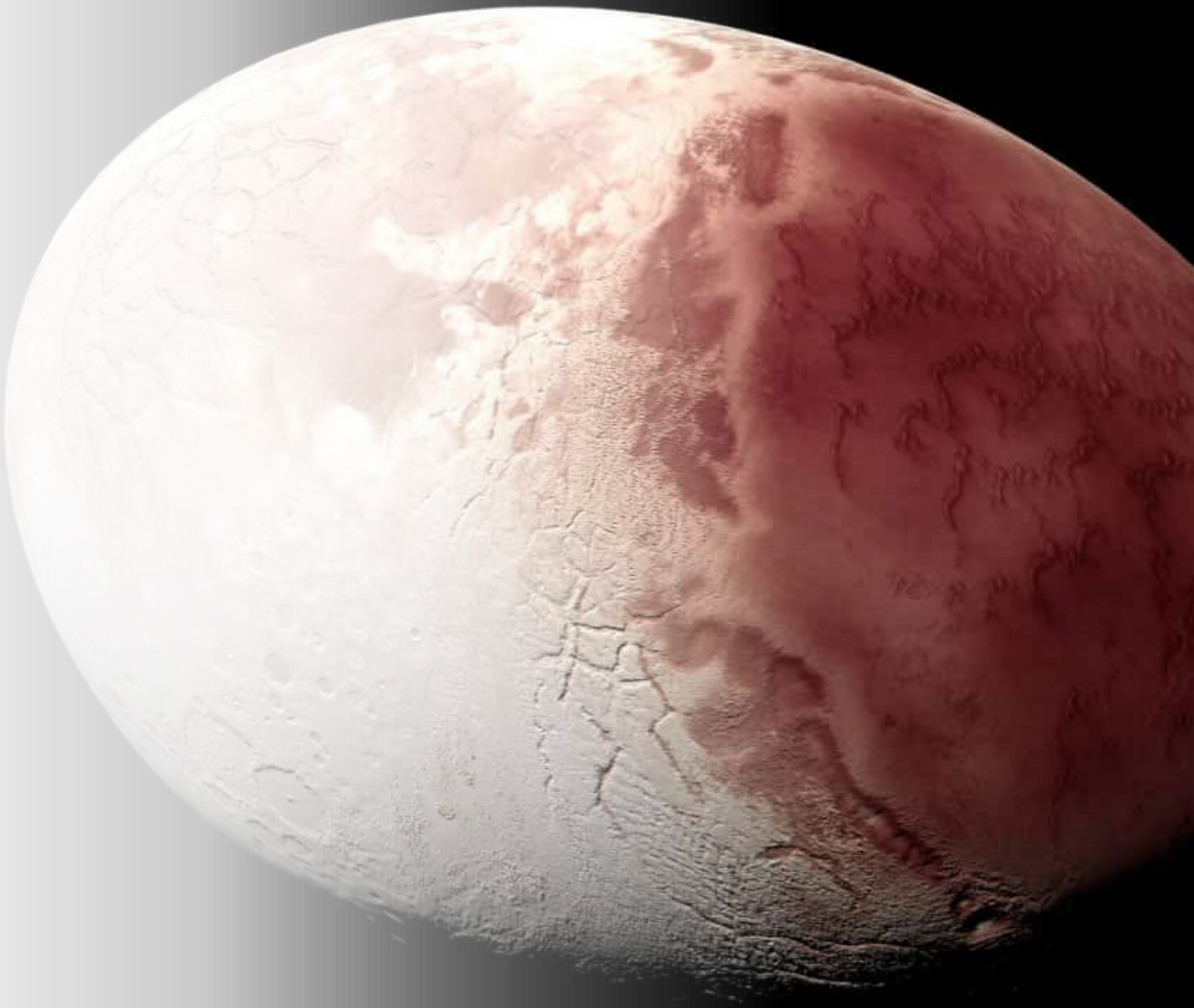
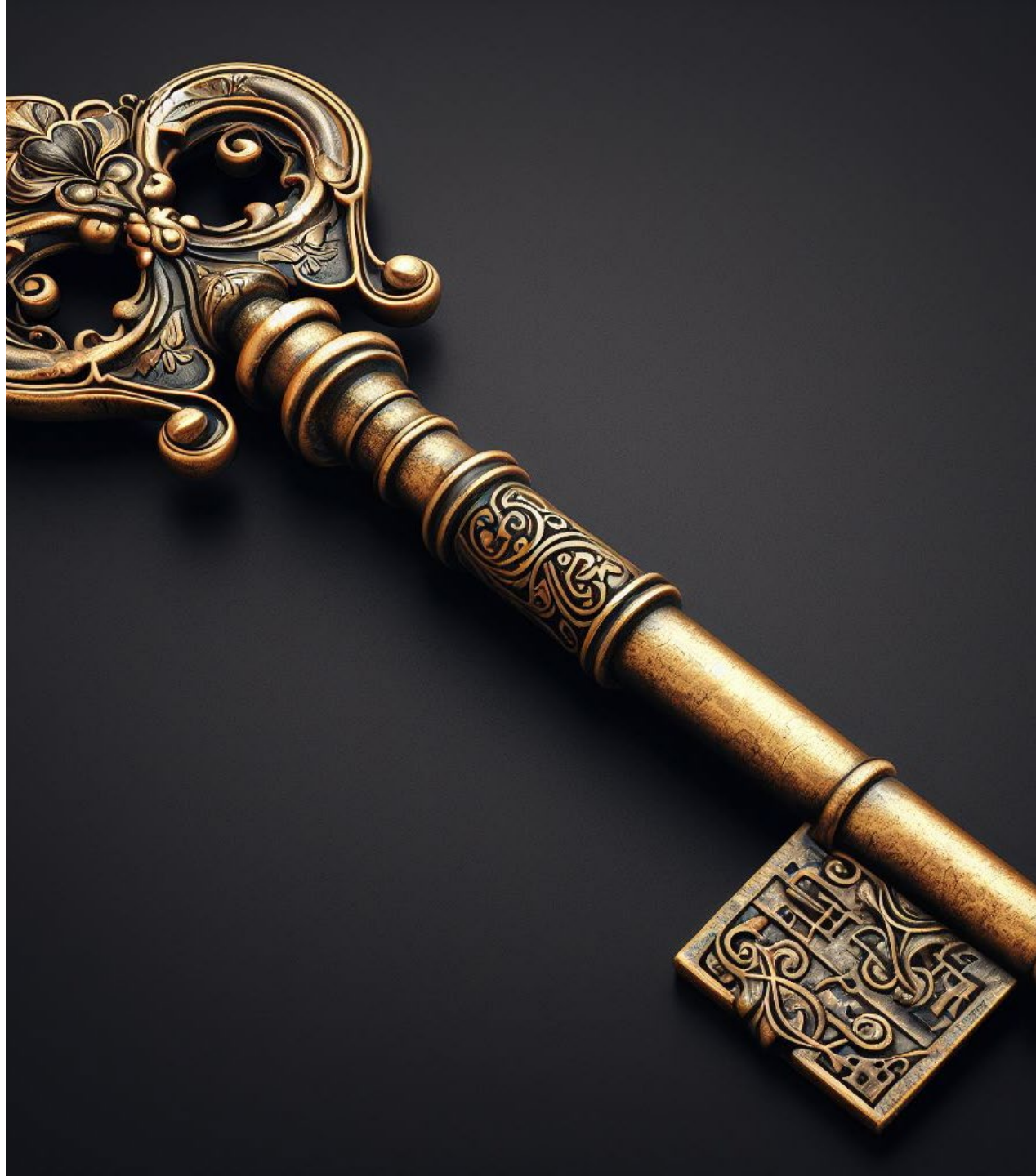
CTD Intro Week 3

JavaScript Objects



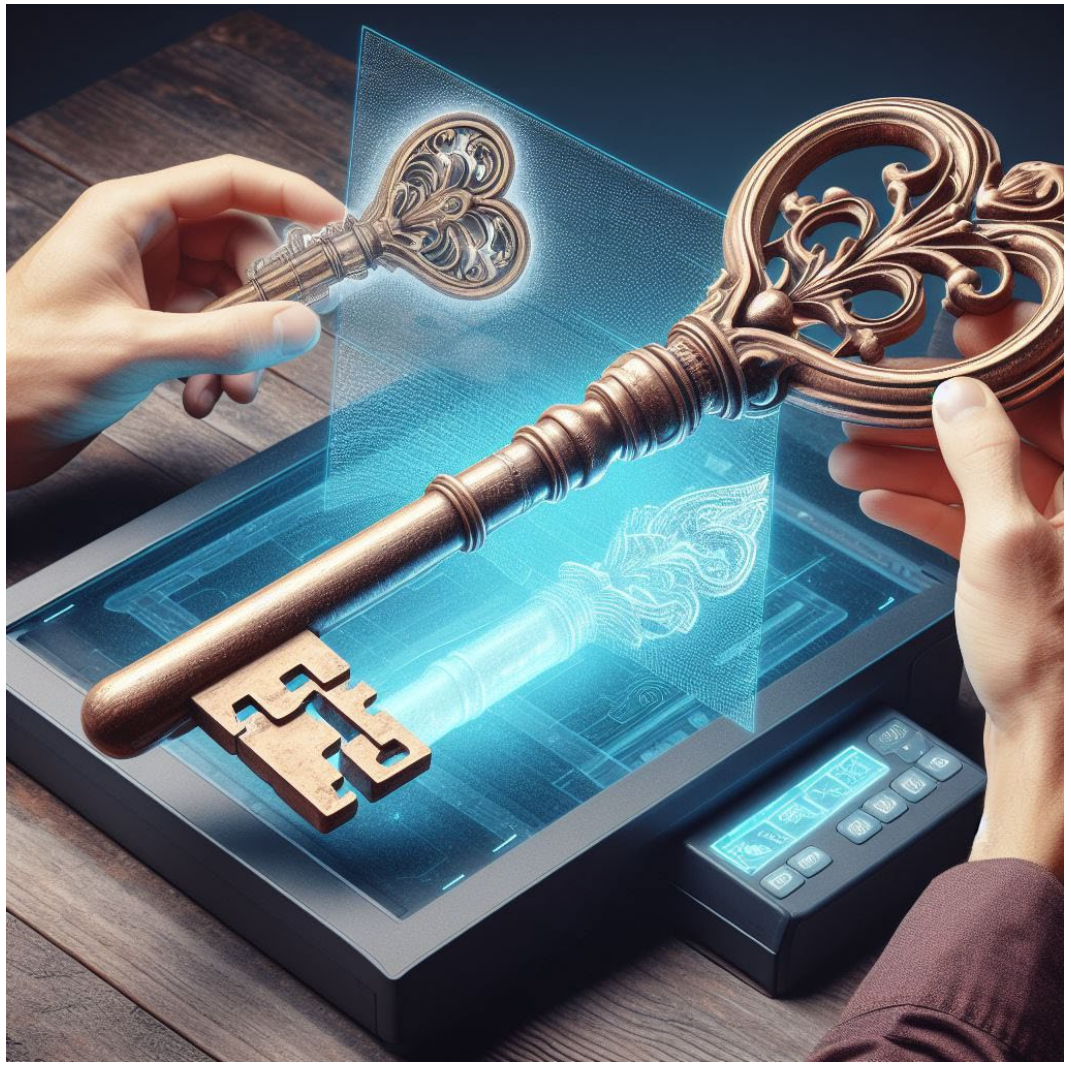
Object Basics

- Another type of array
  - Indexed using a string or Symbol
  - AKA associative array
  - AKA key/value array
- Object literal
  - let newObj = {aKey: “aString”, otherKey: 63};
  - aKey and otherKey are called keys or properties of the object
- Object access and assignment:
  - newObj[“aKey”] // “aString”
  - newObj.aKey // “aString”
  - newObj[“newKey”] = 57; // newObj.newKey is 57
  - newObj.newerKey = 95; //newObj[“newerKey”] is 95
- Test key existence
  - “newKey” in newObj // true
- Remove a property (also called key)
  - delete newObj.aKey
- Loop on keys
  - for (let key in newObj) { … } // “aKey”, “newKey”, “newerKey”
  - OK to use for…in for Objects, not for Arrays (use for…of)



Assignment and Copying

- A variable holds a reference to an object
  - Assigning it to another variable just assigns the reference
    - It's the same object!
  - A const object variable is only a constant reference
    - It can't be re-assigned, but the object content can be changed
- Copying objects
  - Shallow copy (only the top-level properties)
    - `let copiedObj = Object.assign({}, origObj);`
  - Spread operator (also a shallow copy)
    - `let copiedObj = { …origObj };`
    - It also works for arrays and strings
- JSON.stringify() can be used to make deep copies
  - `let deepCopyObj = JSON.parse(JSON.stringify(origObj));`
- Comparison
  - ==, ===, etc. only check whether a variable is the same reference
  - `JSON.stringify(objA) == JSON.stringify(objB) // will do a deep comparison`



Methods and Constructors

- Instance: a specific copy of a specific type of object
- A method is a function which is a property of an object
- this is a special variable which refers to the current object instance
- A Constructor is a function which creates an instance of a specific type of object
  - Constructors are called using new
  - They are named with Uppercase
  - let bigWidget = new Widget(“large”, “purple”);

```
// example of a method
let myWidget = {
    gear: "helical",
    size: "medium",
    report: function() { // a method
                console.log(`gear type: ${this.gear} size: ${this.size}`)
            }
}
// myWidget.report() writes 'gear type: helical size: medium' to the console

// example of a Constructor
function Widget(gear, size) {
    let obj = {
        gear, // shorthand for: gear: gear
        size, // shorthand for: size: size
        report: function() {
            console.log(`gear type: ${this.gear} size: ${this.size}`)
        }
    };
    return obj;
}
let bigWidget = new Widget("toothed", "large");
// bigWidget.report() writes gear type: toothed size: large to the console
```

Complex Data Structures

- JavaScript objects can be any mix of associative, indexed arrays, scalars
  - Very flexible way to represent complex data structures
  - typeof([]) returns ‘object’ // arrays are objects too!
- So flexible and convenient that is has become a data exchange standard
  - JSON (JavaScript Object Notation)
  - Almost all languages support JSON now
  - It’s native to JavaScript!

```
// complex datatype
let fancyDataType = {
    "data": [
        {
            "MainId": 1111,
            "firstName": "Sherlock",
            "lastName": "Homes",
            "categories": [
                {
                    "CategoryID": 1,
                    "CategoryName": "Example"
                }
            ]
        },
        {
            "MainId": 122,
            "firstName": "James",
            "lastName": "Watson",
            "categories": [
                {
                    "CategoryID": 2,
                    "CategoryName": "Example2"
                }
            ]
        }
    ],
    "messages": [], // blank json
    "success": true // boolean value
}
```

Dates and Times

- Unix time (epoch)
  - The number of seconds since 1/1/1970
  - Date.now() returns the number of milliseconds since 1/1/1970
- Date module
  - let today = new Date() creates a date object at the current date and time
  - today.getTime() unix epoch at the time ‘today’ was created
  - getFullYear(), getMonth(), getDate(), getDay()
  - getHours(), getMinutes(), getSeconds(), getMilliseconds()

Object History

- Objects were invented in the Simula language in the 1960's (starting in 1962)
  - A language designed for simulation
  - Introduced the idea of objects with named properties
- Most languages now provide objects in some form
  - python, ruby, java, c++…

Q and A and Demos