CS 4220

- Current Trends in Web Design & Development -

Cydney Auman

AGENDA

Template Literals

Backtick-quoted strings, usually called **template literals** or **template strings** and in using this notation we can do a few more tricks. Not only they we span multiple lines, they can also embed other values.

When you write something inside **${ }** in a template literal, its result will be computed, then converted to a string, and included at that position.

The example below produces: “half of 100 is 50”.

```
`half of 100 is ${100 / 2}`
```

Default Values

In JavaScript, function parameters default to undefined when no value is passed. It's often useful to set a different default value. Default function parameters allow named parameters to be initialized with **default values** if no value or undefined is passed.

```
function multiply(a, b = 1) {
    return a * b;
}
multiply(5, 2);  // 10
multiply(5);     // 5
```

Higher-Order Functions

Functions that operate on other functions, either by taking them as arguments or by returning them, are called **higher-order functions**. Since we have already seen that functions are regular values in JavaScript and that the language support First Class Function, then this is expected that these types of functions exist.

```
function repeat(n, action) {
   for (let i = 0; i < n; i++) {
       action(i);
   }
}
repeat(5, console.log);
```

Array Method: forEach

The **`forEach()`** method executes the provided function once for each array element.

```javascript
const inventory = [1, 5, 7, 3, 2];
inventory.forEach(count => {
   console.log(`Unit: ${count}`)
})
```

Array Method: map

The `map()` method creates a new array populated with the results of calling a provided function on every element in the original array.

```
const squares = [2, 3, 4, 5];
const results = squares.map(n => {
   return n * n;
});
console.log(results);
// [ 4, 9, 16, 25 ]
```

Array Method: filter

The **`filter()`** method creates a new array with all elements that pass the expression implemented in the provided function.

```
const mixtypes = [5, 2, 'a', 4, true, 'b', 'c', 7, false, 8];
const numArray = mixtypes.filter(element => {
   if (typeof element === 'number') {
        return element;
   }
});
console.log(numArray);
// [ 5, 2, 4, 7, 8 ]
```

Array Method: reduce

The **reduce()** method executes a the reducer function provided on each element of the array. The reducer returned value is assigned to the accumulator, whose value is remembered across each iteration throughout the array, and ultimately becomes the final, single resulting value.

```
const sumArray = [1, 2, 3, 4, 5]
const sum = sumArray.reduce((accumulator, element) => {
   return accumulator + element;
}, 0);
console.log(sum); // 15
```

Array Method: concat and ... (spread)

The **`concat()`** method is used to merge two or more arrays. This method does not change the existing arrays, but instead returns a new array.

Spread syntax **`...`** allows an iterable such as an array or string to be expanded in places.

```
const a1 = [1, 2, 3];
const a2 = [2, 3, 4];
const concat = a1.concat(a2);
const spread = [...a1, ...a2];
```

Object Destructuring

The **destructuring assignment** syntax is a JavaScript expression that makes it possible to extract data from objects into distinct variables. A variable can be assigned a default value, in the case that the key unpacked from the object is not defined.

```
const fido = {
    breed: 'border collie',
    colors: ['black', 'white']
}
const { type = 'dog', breed, colors } = fido;
```

```
console.log('Week 03');
console.log('Code Examples');
```