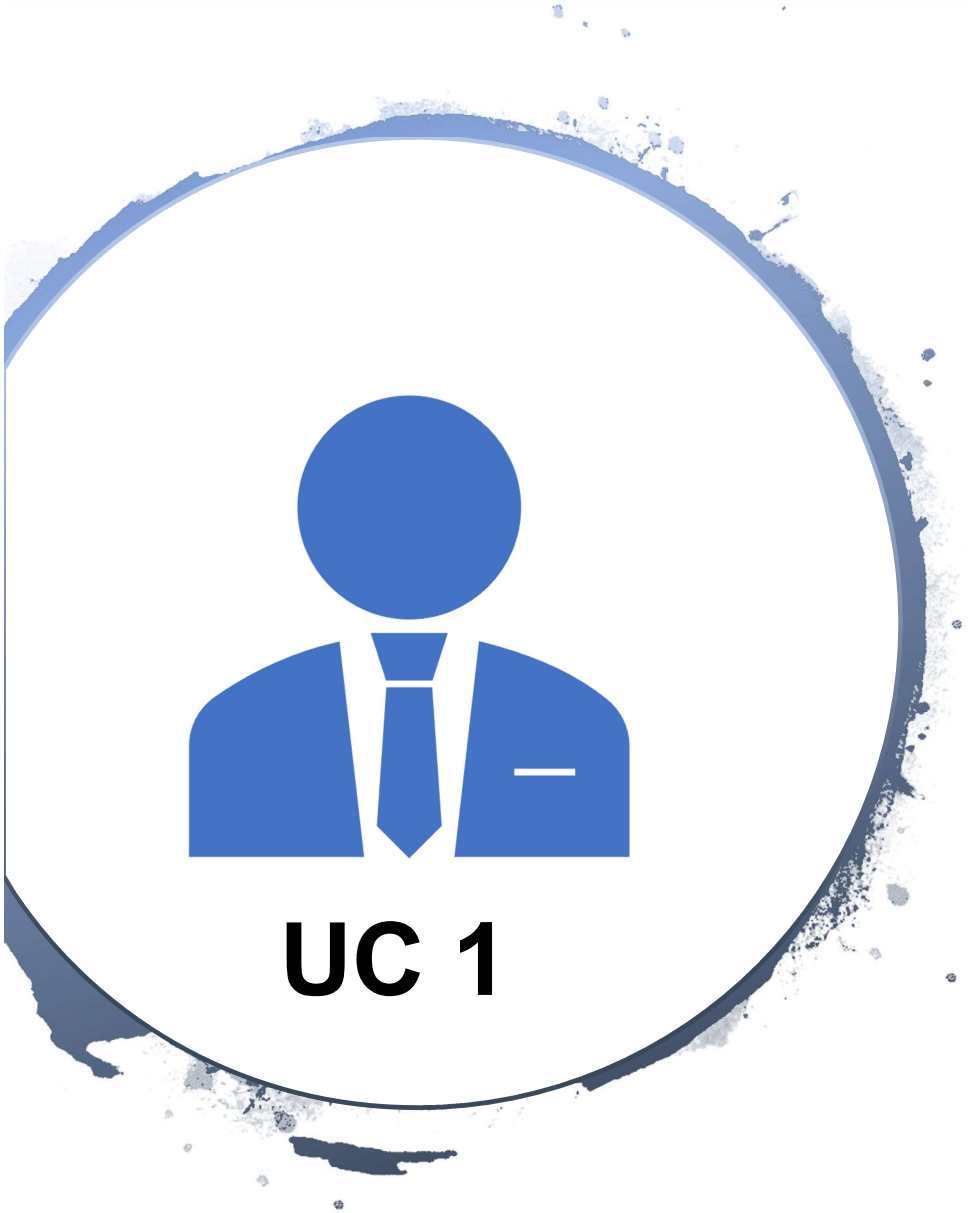
BridgeLabz

Employability Delivered

# HTML, CSS & Javascript

# Add Employee Payroll




BridgeL
Employability Del

# Modify Employee Payroll Class with new Attributes and Getters and Setters

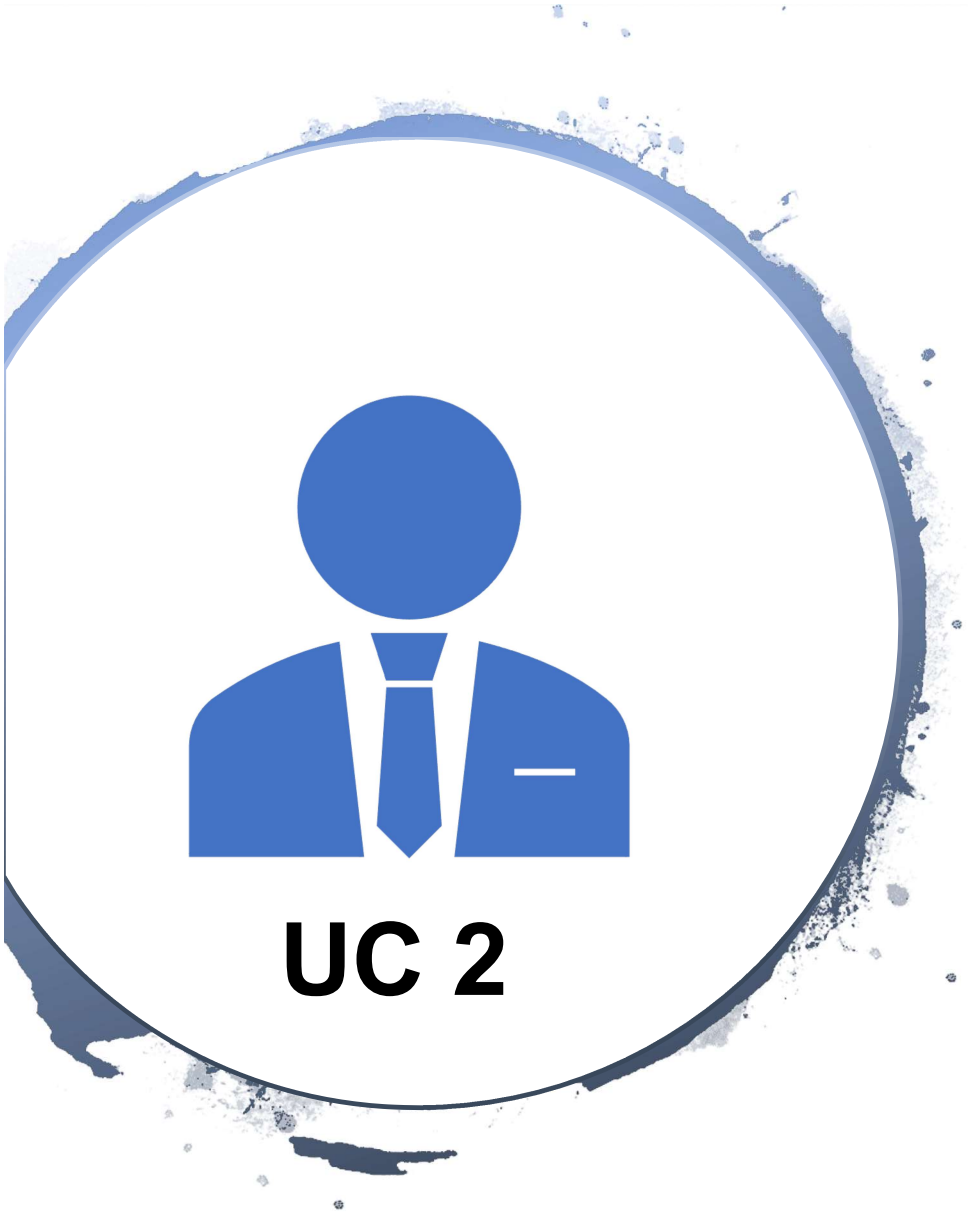
- New Attributes added are Department, Gender, Employee Notes, Profile Pic, etc
- Note – Getters and Setters are used for all properties and Constructor is made default

js > JS EmployeePayroll.js > ...

```
class EmployeePayrollData {

  // getter and setter method
  get id() { return this._id; }
  set id(id) {
    this._id = id;
  }

  get name() { return this._name; }
  set name(name) {
    let nameRegex = RegExp('^[A-Z]{1}[a-zA-Z\\s]{2,}$')
    if (nameRegex.test(name))
      this._name = name;
    else throw 'Name is Incorrect!';
  }

  get profilePic() { return this._profilePic; }
  set profilePic(profilePic) {
    this._profilePic = profilePic;
  }

  get gender() { return this._gender; }
  set gender(gender) {
    this._gender = gender;
  }

  get department() { return this._department; }
  set department(department) {
    this._department = department;
  }

  get salary() { return this._salary; }
  set salary(salary) {
    this._salary = salary;
  }

  get note() { return this._note; }
  set note(note) {
    this._note = note;
  }

  get startDate() { return this._startDate; }
  set startDate(startDate) {
    this._startDate = startDate;
  }

  // method
  toString() {
    const options = { year: 'numeric', month: 'long', day: 'numeric' };
    const empDate = !this.startDate ? "undefined" :
                    this.startDate.toLocaleDateString("en-US", options);
    return "id=" + this.id + ", name='" + this.name + ", gender='" + this.gender +
           ", profilePic='" + this.profilePic + ", department=" + this.department
           ", salary=" + this.salary + ", startDate=" + empDate + ", note=" + this
  }
}
```

BridgeL
Employability Del

# UC 1 – Employee Payroll Class with New Attributes




BridgeL
Employability Del

# Ability to set Event Listeners when Document is loaded so as to

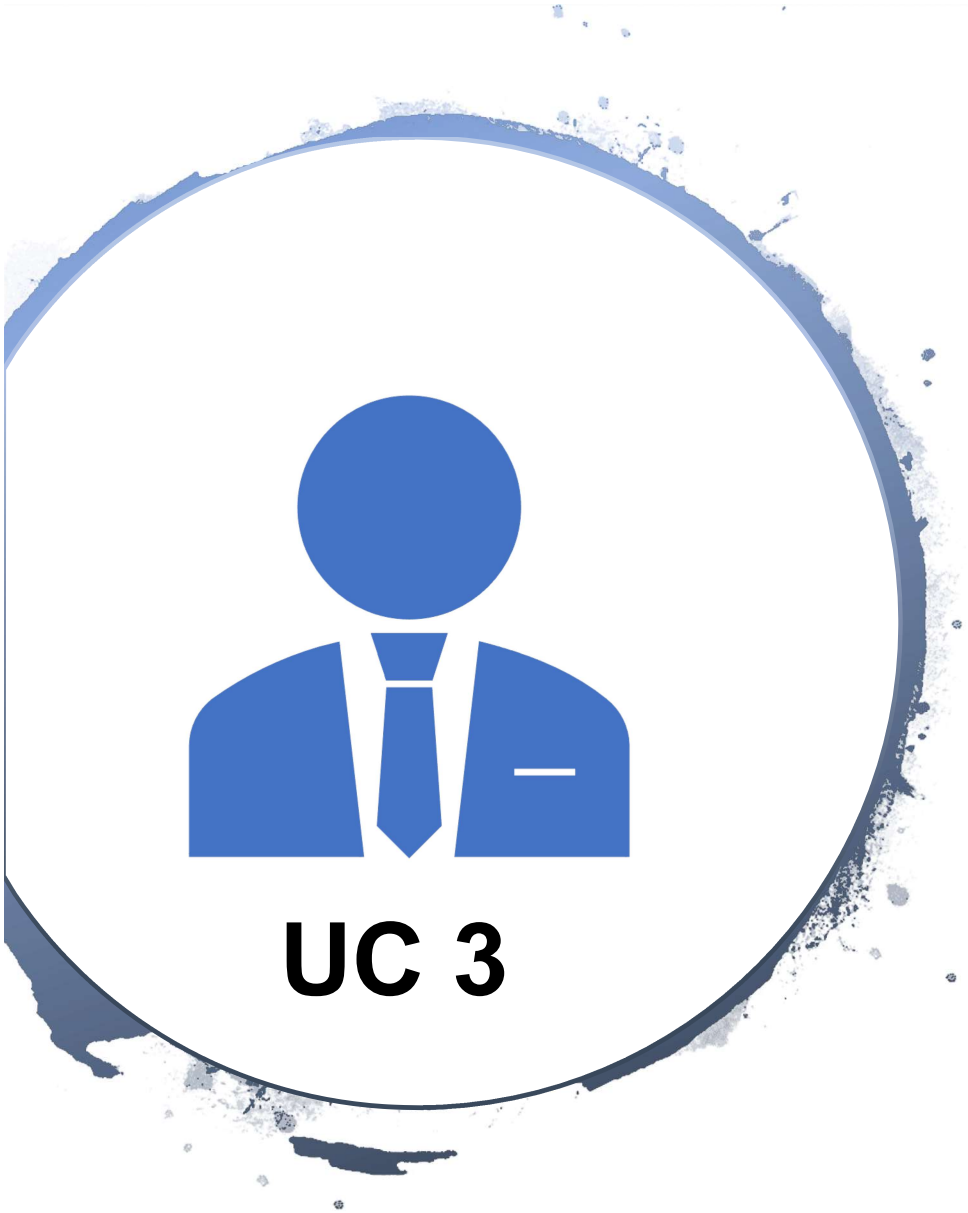
- Set Event Listener on Salary Range to display appropriate value
- Validation of Name and Date

# UC 2 – On Document Load Set Event Listeners


```html
<div id="formId" class="form-content">
  <form class="form" action="#" onreset="resetForm()" onsubmit="save()">
    <!-- UC 2 -->
    <div class="form-head"> Employee Payroll form </div>
    <div class="row-content">
      <label class="label text" for="name">Name</label>
      <input class="input" type="text" id="name" name="name"
             placeholder="Your name.." required>
      <error-output class="text-error" for="text"></error-output>
    </div>
  </form>
</div>
```

```css
error-output {
    margin-left: 10px;
    font-size: 12px;
    font-style: italic;
    color: red;
}
```

```javascript
window.addEventListener('DOMContentLoaded', (event) => {
    const name = document.querySelector('#name');
    const textError = document.querySelector('.text-error')
    name.addEventListener('input', function() {
        if(name.value.length == 0) {
            textError.textContent = "";
            return;
        }
        try {
            (new EmployeePayrollData()).name = name.value;;
            textError.textContent = "";
        } catch (e) {
            textError.textContent = e;
        }
    });

    const salary = document.querySelector('#salary');
    const output = document.querySelector('.salary-output')
    output.textContent = salary.value;
    salary.addEventListener('input', function() {
      output.textContent = salary.value;
    });
});
```




BridgeL
Employability Del

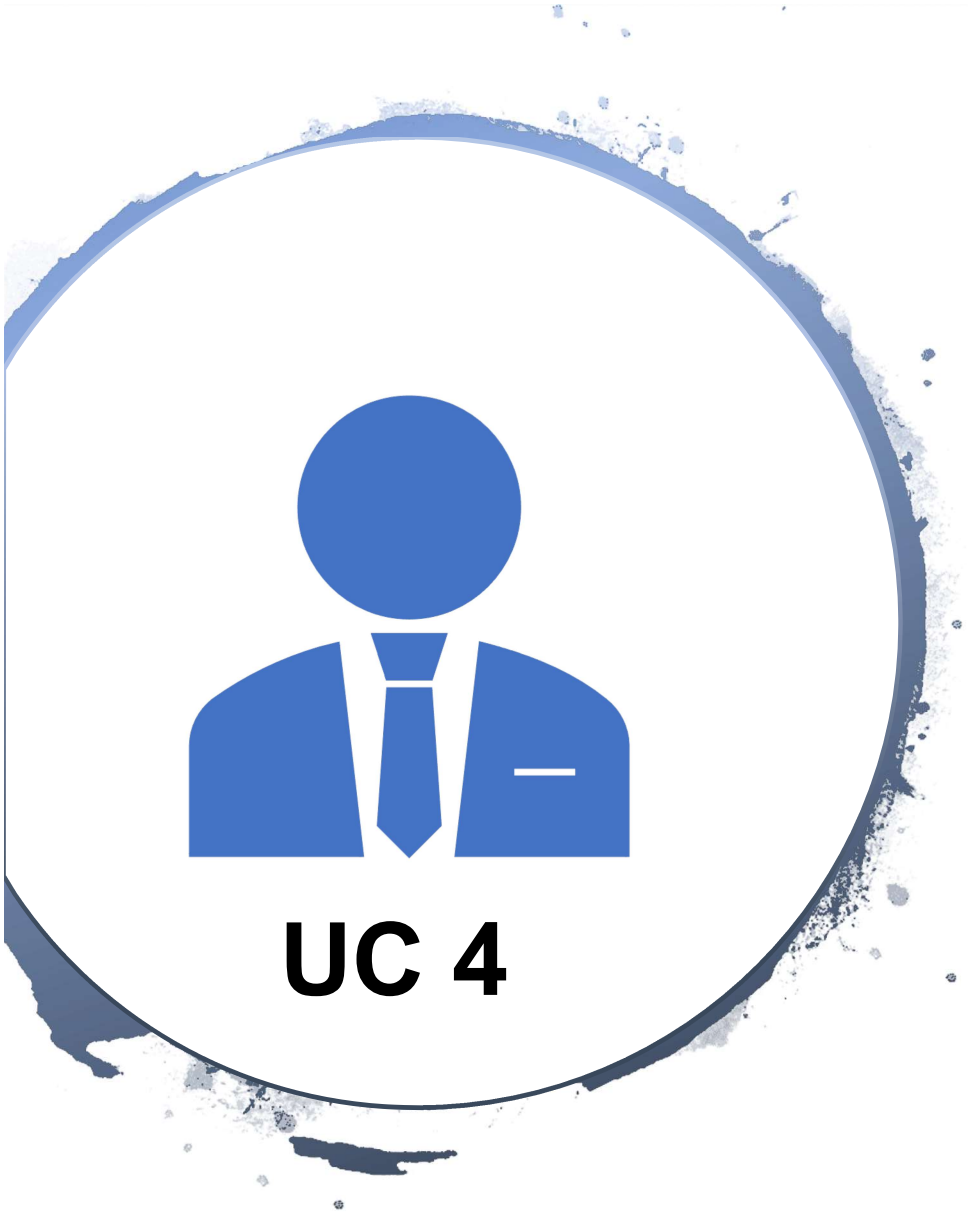
# Ability to create Employee Payroll Object On Save.

- Validation of Name and Date and if failed then set the UI accordingly

# UC 3 – On Save Create Employee Payroll Object


```
const save = () => {
    try {
        let employeePayrollData = createEmployeePayroll();
    } catch (e) {
        return;
    }
}

const createEmployeePayroll = () => {
    let employeePayrollData = new EmployeePayrollData();
    try {
      employeePayrollData.name = getInputValueById('#name');
    } catch (e) {
      setTextValue('.text-error', e);
      throw e;
    }
    employeePayrollData.profilePic = getSelectedValues('[name=profile]').pop();
    employeePayrollData.gender = getSelectedValues('[name=gender]').pop();
    employeePayrollData.department = getSelectedValues('[name=department]');
    employeePayrollData.salary = getInputValueById('#salary');
    employeePayrollData.note = getInputValueById('#notes');
    let date = getInputValueById('#day')+" "+getInputValueById('#month')+" "+
               getInputValueById('#year') ;
    employeePayrollData.date = Date.parse(date);
    alert(employeePayrollData.toString());
    return employeePayrollData;
}

const getSelectedValues = (propertyValue) => {
    let allItems = document.querySelectorAll(propertyValue);
    let selItems = [];
    allItems.forEach(item => {
      if(item.checked) selItems.push(item.value);
    });
    return selItems;
}
```

```
/*
 * 1: querySelector is the newer feature.
 * 2: The querySelector method can be used when selecting by element name,
 *    nesting, or class name.
 * 3: querySelector lets you find elements with rules that can't be
 *    expressed with getElementById
 */
const getInputValueById = (id) => {
    let value = document.querySelector(id).value;
    return value;
}

/*
 * 1: getElementById is better supported than querySelector in older versi
 *    of the browsers.
 * 2: The thing with getElementById is that it only allows to select an
 *    element by its id.
 */
const getInputElementValue = (id) => {
    let value = document.getElementById(id).value;
    return value;
}
```




BridgeL
Employability Del

# Ability to save the Employee Payroll Object to Local Storage.

- Understand the difference between Local Storage, Session Storage and older feature of storing in cookies. Here are good references
- HTML5 Storage
- Quick Guide Local Storage supported Methods

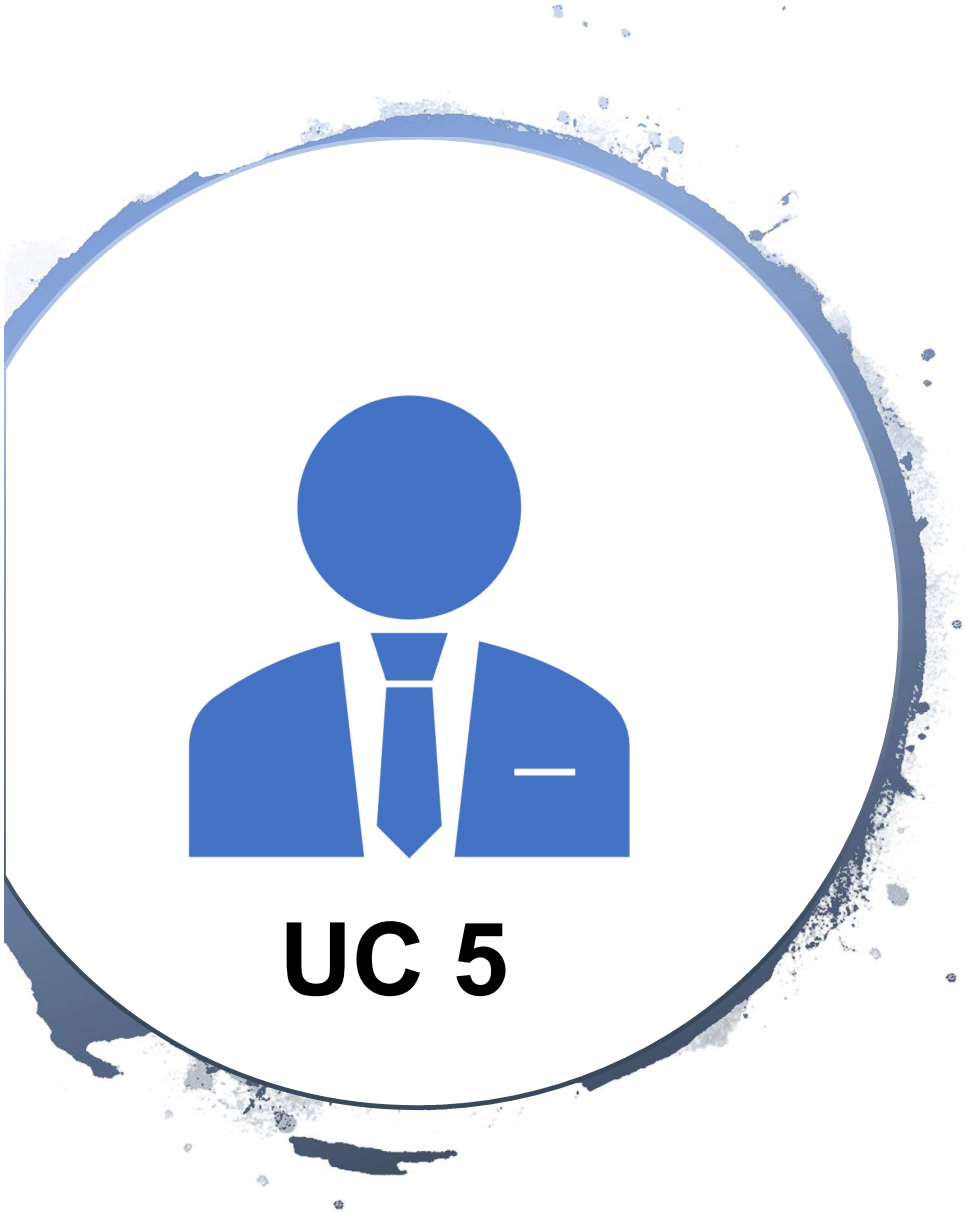
# UC 4 – Saving Employee Payroll to Local Storage


```
const save = () => {
    try {
        let employeePayrollData = createEmployeePayroll();
        createAndUpdateStorage(employeePayrollData);
    } catch (e) {
        return;
    }
}

function createAndUpdateStorage(employeePayrollData){

    let employeePayrollList = JSON.parse(localStorage.getItem("EmployeePayrollList"));

    if(employeePayrollList != undefined){
        employeePayrollList.push(employeePayrollData);
    } else{
        employeePayrollList = [employeePayrollData]
    }
    alert(employeePayrollList.toString());
    localStorage.setItem("EmployeePayrollList", JSON.stringify(employeePayrollList))
}
```

# HTML5 Local Storage


- **Cookies** are small pieces of data which a **server can store in the browser**. The cookie is sent by the browser along with all future HTTP requests to the server that set the cookie. Cookies cannot be bigger than **4KB** in total.
- **HTML5 local storage** is set via JavaScript executed in the browser. HTML5 local storage properties are never sent to any server and can store data up to 5MB.
- The properties set in the HTML5 local storage can only be read by pages from the **same domain** as the page that set the properties, thus maintaining **storage security**.
- Local storage is available in the browser to all windows with the same origin (domain). Data stored in the local storage is also available after the **window has been closed**.
- **Session storage** is available inside the same browser window for as long as the window is open. When the browser window is closed, the session storage associated with that window is **deleted**.

1. `setItem()` : Add key and value to `localStorage`
2. `getItem()` : Retrieve a value by the key from `localStorage`
3. `removeItem()` : Remove an item by key from `localStorage`
4. `clear()` : Clear all `localStorage`
5. `key()` : Passed a number to retrieve nth key of a `localStora`




BridgeL
Employability Del

Ability to reset the form on clicking reset

## UC 5 – Reset the Employee Payroll Form

```
const resetForm = () => {
    setValue('#name','');
    unsetSelectedValues('[name=profile]');
    unsetSelectedValues('[name=gender]');
    unsetSelectedValues('[name=department]');
    setValue('#salary','');
    setValue('#notes','');
    setValue('#day','1');
    setValue('#month','January');
    setValue('#year','2020');
}

const unsetSelectedValues = (propertyValue) => {
    let allItems = document.querySelectorAll(propertyValue);
    allItems.forEach(item => {
        item.checked = false;
    });
}

const setTextValue = (id, value) => {
    const element = document.querySelector(id);
    element.textContent = value;
}

const setValue = (id, value) => {
    const element = document.querySelector(id);
    element.value = value;
}
```

# References


- HTML Basics
- CSS Basics
- HTML / CSS in a nutshell
- HTML Element Ref
- Design webpage structure
- Flex Box
- A Guide to Flex Box
- Flex Layout Example
- Flex Live Demos
- Flex Try it out
- HTML-CSS-JS Try it out
- Open in Mobile HTML Form
- HTML Form Fields
- HTML 5 Input Fields
- Complete Guide to CSS Media Queries
- HTML5 Storage
- Quick Guide Local Storage supported Methods

BridgeLabz

Employability Delivered

# Thank You