

F28WP Web programming


Lab Sheet 2

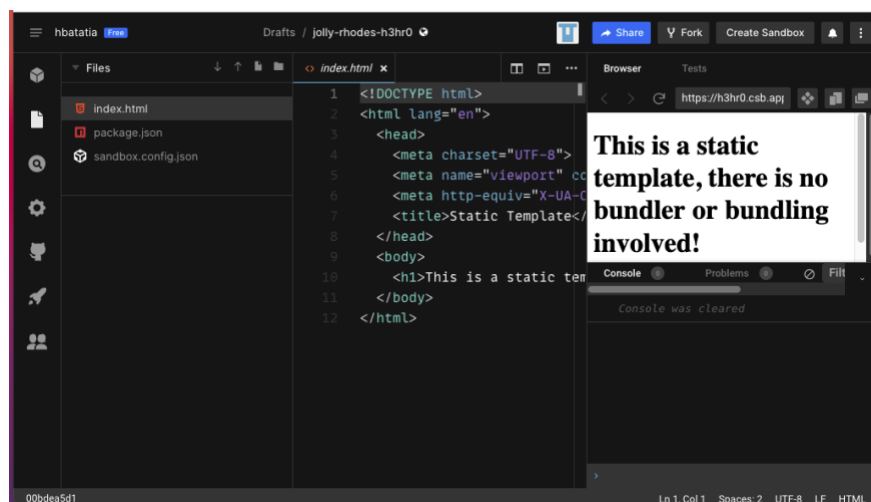
JavaScript with DOM API

Activities in this lab session concentrate on client-side programming using HTML, CSS and JavaScript with Objects, Arrays and DOM API. It consists in developing a simple (single user) game without any server interactions. Two characters compete on a game board. One character moves randomly and the second is controlled by the user through keyboard keys (up, down, left, right). The game is configurable to add more characters and control their speed. Scores are displayed continuously.

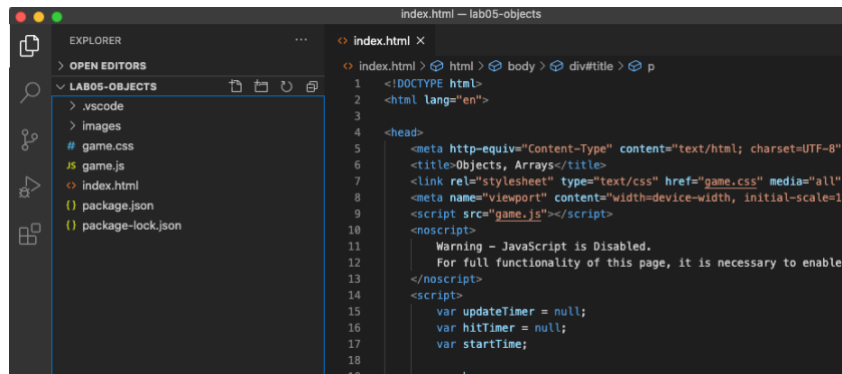
1. Tools (0 mark)

This lab requires only a code editor and a browser. Any development tool can be used for this purpose. However, to organize properly the development, it is recommended to use an integrated development environment (IDE). Any online IDE can be used. However, **codesandbox.io is recommended**. You can also use locally installed tools from the computers in the lab rooms or on your own computer. Visual Studio code (VSC) is installed in most lab rooms and can easily be installed on your own computer.

If you are working with codesandbox, sign in and create a sandbox using the static () template.

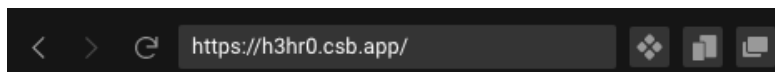


If you are using locally installed tools, create a folder for your project and open it under Visual studio code.



Your project must be stored in a GitHub repository. If you are working with codesandbox, online, click on the GitHub button (🔗) to create a repository for your project. If you are working locally, create a new GitHub repository and link it to your project.

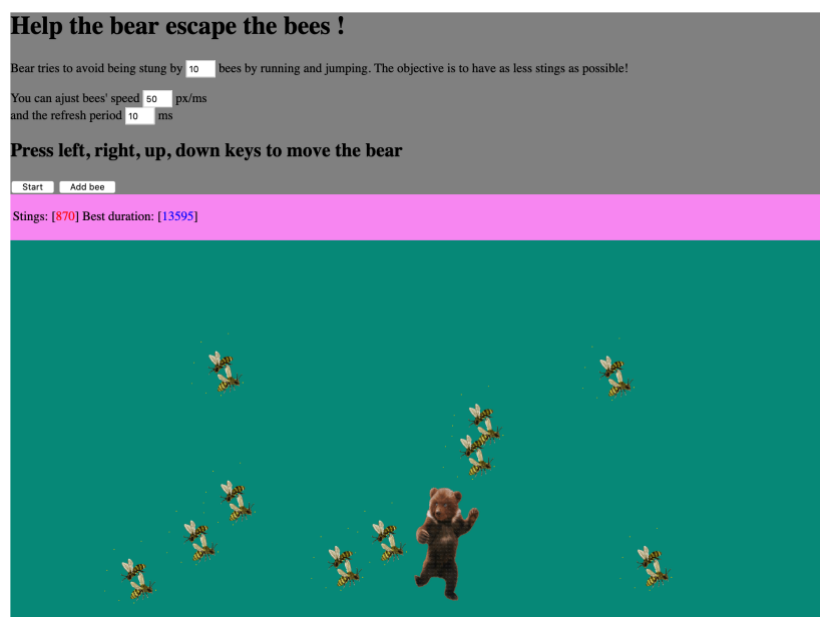
For testing your application, on codesandbox, click on reload button (🔄) or open the page in a new tab (📄).



If you are working locally, open the HTML file in any web browser. From Visual Studio Code, you can create a shortcut for opening the HTML file in the default browser by installing the extension (open in browser) and clicking on Alt + b on Windows and Linux (or option + b on MacOS).

2. Specifications (0 mark)

We would like to create an interactive game where a character is controlled by the player to achieve an objective. **It promotes object programming with JavaScript and the use of the DOM API.** The window of the game has the following shape



The characters are bees and a bear. The bees fly randomly in all directions to attack the bear. The bear is controlled by the user and can move up, down, left, and right while staying in the game board to avoid being bitten by bees.

The objective of the player is to minimize the number of stings. The winner is the player who manages to have less stings.

The user must be able to adjust, on the run, the speed of the bees and the frequency of the game update loop. They can also restart the game and set any the number of bees to play with. Bees appear at random positions in the board and move continuously and randomly in the four directions.

The application should detect any sting (overlap between a bee and the bear images). The number of stings is displayed continuously. The duration between two stings is also calculated. The longest period without stings is kept as the best score. The largest duration is displayed and refreshed every time it changes. The user can add bees at any time without restarting the game.

3. Create the web page of the game (1 mark)

Using HTML and CSS, start creating a web page that looks like the above, with preferably three div element:

```
<body>

  <div id="title">

  </div>
  <div id="scores">

  </div>
  <div id="board">

  </div>
</body>
```

The **title** div contains rules of the game and configuration input fields, as well as the control buttons.

The **scores** div shows the number of number of stings and the best duration.

The **board** div contains initially the bear image, added manually; whereas the bee image elements are created and added dynamically depending on the number given by the under.

CSS rules are used to set up the size, color, and position properties of the elements. Use the **resources 1 and 2** for the images, **resource 3** for the html file, and **resource 4** for the CSS file.

Add the bear image to the board div

```

```

Open your page in a browser. No interaction is possible yet. We need to do JavaScript programming to add dynamics to the page.

4. Bear motion (2 marks)

We first create JavaScript code to control the movement of the bear.
Create the file **game.js** and link it to the HTML file by adding the following to the head section:

```
<script src="game.js"></script>
<noscript>
  Warning - JavaScript is Disabled.
  For full functionality of this page, it is necessary to enable JavaScript.
</noscript>
```

To control the bear position in the game board, we create a class that holds the HTML attributes of the bear image. We also add two functions (or methods):

- move() to move the bear by dx and dy steps in the horizontal and vertical directions.
- display() to display the bear at the new position

Add the following code to the game.js file:

```
function Bear() {
  this.dBear = 100;
  this.htmlElement = document.getElementById("bear");
  this.id = this.htmlElement.id;
  this.x = this.htmlElement.offsetLeft;
  this.y = this.htmlElement.offsetTop;

  this.move = function(xDir, yDir) {
    this.x += this.dBear * xDir;
    this.y += this.dBear * yDir;
    this.display();
  };

  this.display = function() {
    this.htmlElement.style.left = this.x + "px";
    this.htmlElement.style.top = this.y + "px";
    this.htmlElement.style.display = "absolute";
  };
}
```

The attribute **dBear** is the step (in pixels) made by bear when the user clicks an arrow on the keyboard. notice the use of the DOM API in the move function and in the display function.

Testing the page does not bring any change yet. We need to instantiate the Bear object. For this purpose, we need to create a global variable when the page first loads. This variable declaration is done in the head section of the HTML file:

```
<script>
  var bear;
```

```
</script>
```

The actual instantiation is done in a `start()` function (in the `game.js` file):

```
function start() {  
    //create bear  
    bear = new Bear();  
}
```

To invoke the `start()` function, we add `onload` event to the `body` tag in the HTML page:

```
<body onload="start();">
```

This will invoke the `start()` function when the page finishes to load.

Now, to be able to control the movement of the bear, we create an event handler for keyboard events (Up, Down, Left, Right) – Notice the call to the function `move()` of the bear object:

```
// Handle keyboard events  
// to move the bear  
function moveBear(e) {  
    //codes of the four keys  
    const KEYUP = 38;  
    const KEYDOWN = 40;  
    const KEYLEFT = 37;  
    const KEYRIGHT = 39;  
  
    if (e.keyCode == KEYRIGHT) {  
        bear.move(1, 0)  
    } // right key  
    if (e.keyCode == KEYLEFT) {  
        bear.move(-1, 0)  
    } // left key  
  
    if (e.keyCode == KEYUP) {  
        bear.move(0, -1)  
    } // up key  
    if (e.keyCode == KEYDOWN) {  
        bear.move(0, 1)  
    } // down key  
}
```

The bear does not move yet! We need to associate the event handler to the bear image (the `IMG` tag). We do this in the `start()` function using the DOM API function `addEventListener()`:

```
function start() {  
    //create bear  
    bear = new Bear();
```

```
// Add an event listener to the keypress event.
document.addEventListener("keydown", moveBear, false);
}
```

Now, the user can move the bear. However, it goes beyond the limits of the game board! We need to fit the bear to the board limits. For this, we add the following function in the bear class (notice again the use of the DOM API):

```
this.fitBounds = function() {
    let parent = this.htmlElement.parentElement;
    let iw = this.htmlElement.offsetWidth;
    let ih = this.htmlElement.offsetHeight;
    let l = parent.offsetLeft;
    let t = parent.offsetTop;
    let w = parent.offsetWidth;
    let h = parent.offsetHeight;
    if (this.x < 0) this.x = 0;
    if (this.x > w - iw) this.x = w - iw;
    if (this.y < 0) this.y = 0;
    if (this.y > h - ih) this.y = h - ih;
};
```

We call this function from the move() method:

```
this.move = function(xDir, yDir) {
    this.fitBounds(); //we add this instruction to keep bear within board
    this.x += this.dBear * xDir;
    this.y += this.dBear * yDir;
    this.display();
};
```



Create an input field for the user to enter the speed (dBear) of the bear. Make the necessary changes to read the value entered by the user to modify dBear.

Hint: add a method setSpeed in the bear class. In the start() method, associate setSpeed() to the event “onChange:” of the input tag.

5. Adding bees to the board (1 mark)

The Bee class allows creating and controlling instances of bee objects (IMG tags in the HTML file). We first add the Bee class to game.js:

```
class Bee {
    constructor(beeNumber) {
        //the HTML element corresponding to the IMG of the bee
        this.htmlElement = createBeeImg(beeNumber);
    }
}
```

```

//its HTML ID
this.id = this.htmlElement.id;

//the left position (x)
this.x = this.htmlElement.offsetLeft;

//the top position (y)
this.y = this.htmlElement.offsetTop;


this.move = function(dx, dy) {
    //move the bees by dx, dy
    this.x += dx;
    this.y += dy;
    this.display();
};


this.display = function() {
    //adjust position of bee and display it
    this.fitBounds(); //add this to adjust to bounds
    this.htmlElement.style.left = this.x + "px";
    this.htmlElement.style.top = this.y + "px";
    this.htmlElement.style.display = "block";
};


this.fitBounds = function() {
    //check and make sure the bees stays in the board space
    let parent = this.htmlElement.parentElement;
    let iw = this.htmlElement.offsetWidth;
    let ih = this.htmlElement.offsetHeight;
    let l = parent.offsetLeft;
    let t = parent.offsetTop;
    let w = parent.offsetWidth;
    let h = parent.offsetHeight;

    if (this.x < 0)
        this.x = 0;
    if (this.x > w - iw)
        this.x = w - iw;
    if (this.y < 0)
        this.y = 0;
    if (this.y > h - ih)
        this.y = h - ih;

```

```

    };
  }
}

```

The class has attributes of the bee IMG element from the HTML file and methods to move and display the bee (like the bear). However, the IMG element must be created and added to the web page, precisely in the board div. This is done by calling the createBeImg() function, which create the IMG element, add it to the DOM tree, and sets its initial position randomly.

```

function createBeImg(wNum) {
    //get dimension and position of board div
    let boardDiv = document.getElementById("board");
    let boardDivW = boardDiv.offsetWidth;
    let boardDivH = boardDiv.offsetHeight;
    let boardDivX = boardDiv.offsetLeft;
    let boardDivY = boardDiv.offsetTop;
    //create the IMG element
    let img = document.createElement("img");
    img.setAttribute("src", "images/bee.gif");
    img.setAttribute("width", "100");
    img.setAttribute("alt", "A bee!");
    img.setAttribute("id", "bee" + wNum);
    img.setAttribute("class", "bee"); //set class of html tag img
    //add the IMG element to the DOM as a child of the board div
    img.style.position = "absolute";
    boardDiv.appendChild(img);
    //set initial position
    let x = getRandomInt(boardDivW);
    let y = getRandomInt(boardDivH);
    img.style.left = (boardDivX + x) + "px";
    img.style.top = (y) + "px";
    //return the img object
    return img;
}

```



Write the missing function getRandomInt(max) that generates and returns a random integer between 0 and max. **Hint:** use the Math object provided by JavaScript.

To add the number of bees specified by the user in the input field (nbBees), we create a global variable (**var bees;**) in the <script> tag I the header section of the HTML:

```

<script>
    var bear;
    var bees;

```



```
</script>
```

The actual creation of the bees is done in the method makeBees():

```
function makeBees() {  
    //get number of bees specified by the user  
    let nbBees = document.getElementById("nbBees").value;  
    nbBees = Number(nbBees); //try converting the content of the input to a number  
    if (isNaN(nbBees)) { //check that the input field contains a valid number  
        window.alert("Invalid number of bees");  
        return;  
    }  
    //create bees  
    let i = 1;  
    while (i <= nbBees) {  
        var num = i;  
        var bee = new Bee(num); //create object and its IMG element  
        bee.display(); //display the bee  
        bees.push(bee); //add the bee object to the bees array  
        i++;  
    }  
}
```

For this to work, we need to call makeBees() from the start() function:

```
function start() {  
    //create bear  
    bear = new Bear();  
    // Add an event listener to the keypress event.  
    document.addEventListener("keydown", moveBear, false);  
    //create new array for bees  
    bees = new Array();  
    //create bees  
    makeBees();  
}
```

Test your application! You should see bees added to the board. However, they do not move yet (the local motion you see is built in the GIF image and it is not due to JavaScript).

6. Animate the bees (2 marks)

We first write a function moveBees() that displaces each bee to a random location (dx, dy). Notice the use of the speedBees input value to control the speed of the motion:

```
function moveBees() {  
    //get speed input field value
```

```

let speed = document.getElementById("speedBees").value;
//move each bee to a random location
for (let i = 0; i < bees.length; i++) {
    let dx = getRandomInt(2 * speed) - speed;
    let dy = getRandomInt(2 * speed) - speed;
    bees[i].move(dx, dy);
}
}

```

To get the bees to move continuously, we will use a timer.

```

function updateBees() { // update loop for game
    //move the bees randomly
    moveBees();
    //use a fixed update period
    let period = 10; //modify this to control refresh period
    //update the timer for the next move
    updateTimer = setTimeout('updateBees()', period);
}

```

The updateTimer variable must be declared global in the <script> tag in the HTML file. The initial call to updateBees() is done in the start() function:



Change the code in updateBees() to use the input field periodTimer instead of the fixed period (10).

7. Count the stings (1 mark)

We consider that whenever a bee image intersects with the bear image a sting has happened. We would like to count the stings and display the number in the “hits” span element of the HTML file.

The isHit() method checks if two elements overlap

```

function isHit(defender, offender) {
    if (overlap(defender, offender)) { //check if the two image overlap
        let score = hits.innerHTML;
        score = Number(score) + 1; //increment the score
        hits.innerHTML = score; //display the new score
    }
}

```

It uses the function overlap() that determines the intersection between the two elements

```

function overlap(element1, element2) {
    //consider the two rectangles wrapping the two elements
    //rectangle of the first element

```

```

left1 = element1.htmlElement.offsetLeft;
top1 = element1.htmlElement.offsetTop;
right1 = element1.htmlElement.offsetLeft + element1.htmlElement.offsetWidth;
bottom1 = element1.htmlElement.offsetTop + element1.htmlElement.offsetHeight;
//rectangle of the second element
left2 = element2.htmlElement.offsetLeft; //e2x
top2 = element2.htmlElement.offsetTop; //e2y
right2 = element2.htmlElement.offsetLeft + element2.htmlElement.offsetWidth;
bottom2 = element2.htmlElement.offsetTop + element2.htmlElement.offsetHeight;
//calculate the intersection of the two rectangles
x_intersect = Math.max(0, Math.min(right1, right2) - Math.max(left1, left2));
y_intersect = Math.max(0, Math.min(bottom1, bottom2) - Math.max(top1, top2));
intersectArea = x_intersect * y_intersect;
//if intersection is nil no hit
if (intersectArea == 0 || isNaN(intersectArea)) {
    return false;
}
return true;
}

```

To count the stings, we call the isHit() method when we move each bee in moveBees():

```

function moveBees() {
    //get speed input field value
    let speed = document.getElementById("speedBees").value;
    //move each bee to a random location
    for (let i = 0; i < bees.length; i++) {
        let dx = getRandomInt(2 * speed) - speed;
        let dy = getRandomInt(2 * speed) - speed;
        bees[i].move(dx, dy);
        isHit(bees[i], bear); //we add this to count stings
    }
}

```



Change the code to stop the game and display “Game over!”, when the number of stings reaches 1000.

Hint: in the updateBees() method, check the score before updating the timer. Use clearTimeout() to stop the timer.

8. Calculating the longest escape duration (1 mark)

In addition to the number of stings, we would like to measure and display the longest duration the player manages to hold between two consecutive stings. For this, we modify the isHit() method by adding the following code bloc:

```
function isHit(defender, offender) {
    if (overlap(defender, offender)) { //check if the two image overlap
        let score = hits.innerHTML;
        score = Number(score) + 1; //increment the score
        hits.innerHTML = score; //display the new score
        //calculate longest duration
        let newStingTime = new Date();
        let thisDuration = newStingTime - lastStingTime;
        lastStingTime = newStingTime;
        let longestDuration = Number(duration.innerHTML);
        if (longestDuration === 0) {
            longestDuration = thisDuration;
        } else {
            if (longestDuration < thisDuration) longestDuration = thisDuration;
        }
        document.getElementById("duration").innerHTML = longestDuration;
    }
}
```

You need to declare the variable lastStringTime global in the HTML file (within the script tag in the header). This variable must be initialised in the start() method

```
//take start time
lastStingTime = new Date();
```



Modify the code to count the longest duration only after the first move of the bear.
Hint: initialise the lastStringTime when a keyDown event happens...

9. Additional features (2 marks for any feature added)

Add the following features to you game:

- Create a “restart” button in the HTML file to re-initialise the game.
Hint: use onclick to call start(). Use DOM API to reset (set to zero) the score and the duration, clear the timer, clear the bees array by deleting all bees. This operations should be done in the beginning of the start() method.
- Create an “add bee” button that increases the number of bees by one.
Hint: write addBee() function that adds one to the number of bees, create a new bee, display it and add it to the array of bees.
- Modify the JavaScript code to use jQuery instead of the DOM API throughout the application

Resources

Resource 1: bear image

<https://github.com/hbatatia/lab02-dom-resources/blob/main/images/bear.gif>

Resource 2: bee image

<https://github.com/hbatatia/lab02-dom-resources/blob/main/images/bee.gif>

Resource 3: html file

<https://github.com/hbatatia/lab02-dom-resources/blob/main/index.html>

Resource 4: css file

<https://github.com/hbatatia/lab02-dom-resources/blob/main/game.css>

IMPORTANT: Each lab must be **demonstrated** in one of the allocated lab sessions (either virtually or face-to-face). After you have demonstrated your lab work you should submit a copy of your work on CANVAS as evidence (your mark is allocated/given for the demonstration session). The submitted copy, should be a short report and include, your name, date, which lab, who you demonstrated to, and any screenshots/code/links.