Đồ họa

-->

Tuần 4

Giảng viên: Trần Đức Minh

Nội dung bài giảng

- Xử lý lệnh trong Javascript
- Cơ bản về xử lý sự kiện trong Javascript
- Các phương pháp nắm bắt sự kiện
- Sự kiện của các đối tượng cơ bản
- Các ví dụ

Xử lý lệnh trong Javascript

- Javascript là **ngôn ngữ đơn luồng** nghĩa là các câu lệnh bên trong Javascript **không được thực hiện song song** mà **thực hiện từng dòng một** (đồng bộ - synchronous).
- Tuy nhiên, các **câu lệnh không đồng bộ** (asynchronous) trong Javascript được **thực hiện sau** khi các câu lệnh đồng bộ đã thực hiện xong.
- Một số dạng lệnh không đồng bộ trong Javascript
  - setTimeout
  - ajax
  - promise
  - Các sự kiện xảy ra bên trong chương trình (click, resize, ...)

Xử lý lệnh trong Javascript

- Xét ví dụ sau với setTimeout là lệnh không đồng bộ:

```
console.log('Message 1');
// In thông điệp sau 100 milli giây
setTimeout(function() {
    console.log('Message 2');
}, 100);
console.log('Message 3');
```

- Kết quả chương trình

```
Message 1
Message 3
Message 2
```

Hàm Callback

- Trong Javascript, **hàm là đối tượng** và ta có thể truyền đối tượng làm đối số cho hàm. Tức là ta có thể **đặt một hàm làm đối số của một hàm khác**.
  - Hàm được đặt làm đối số của một hàm khác được gọi là **hàm callback**.
- Ví dụ:

```
function showInfo(callbackFunc) {
    callbackFunc();   // Lời gọi hàm callbackFunc
}
```

Tác dụng của hàm Callback

- Do trong Javascript có nhiều lệnh không đồng bộ nên hàm Callback sẽ giúp điều hướng chương trình theo đúng ý muốn của người lập trình.
- Cách **sử dụng hàm Callback**
  - Đặt hàm Callback làm đối số của một hàm khác.
  - Thực hiện gọi hàm Callback bên trong hàm khác sau khi đã hoàn thành một số nhiệm vụ (lệnh).

Tác dụng của hàm Callback

- Ví dụ:

```
console.log('Message 1');


function second(callbackFunc) {
    setTimeout(() => {
        console.log('Message 2');
        callbackFunc();
    }, 100);
}

function third() {
    console.log('Message 3');
}
```

Các dạng hàm callback khác

- Khởi tạo một đối tượng hàm
  - Ví dụ:

```
const message = function() {
    console.log("Thong bao nay se duoc hien thi sau 3 giay !!!");
}

setTimeout(message, 3000);
```

Các dạng hàm callback khác

- Sử dụng **hàm vô danh** (anonymous)
  - Định nghĩa một hàm trực tiếp trên đối số của một hàm khác thay vì gọi đến nó.
  - Ví dụ:

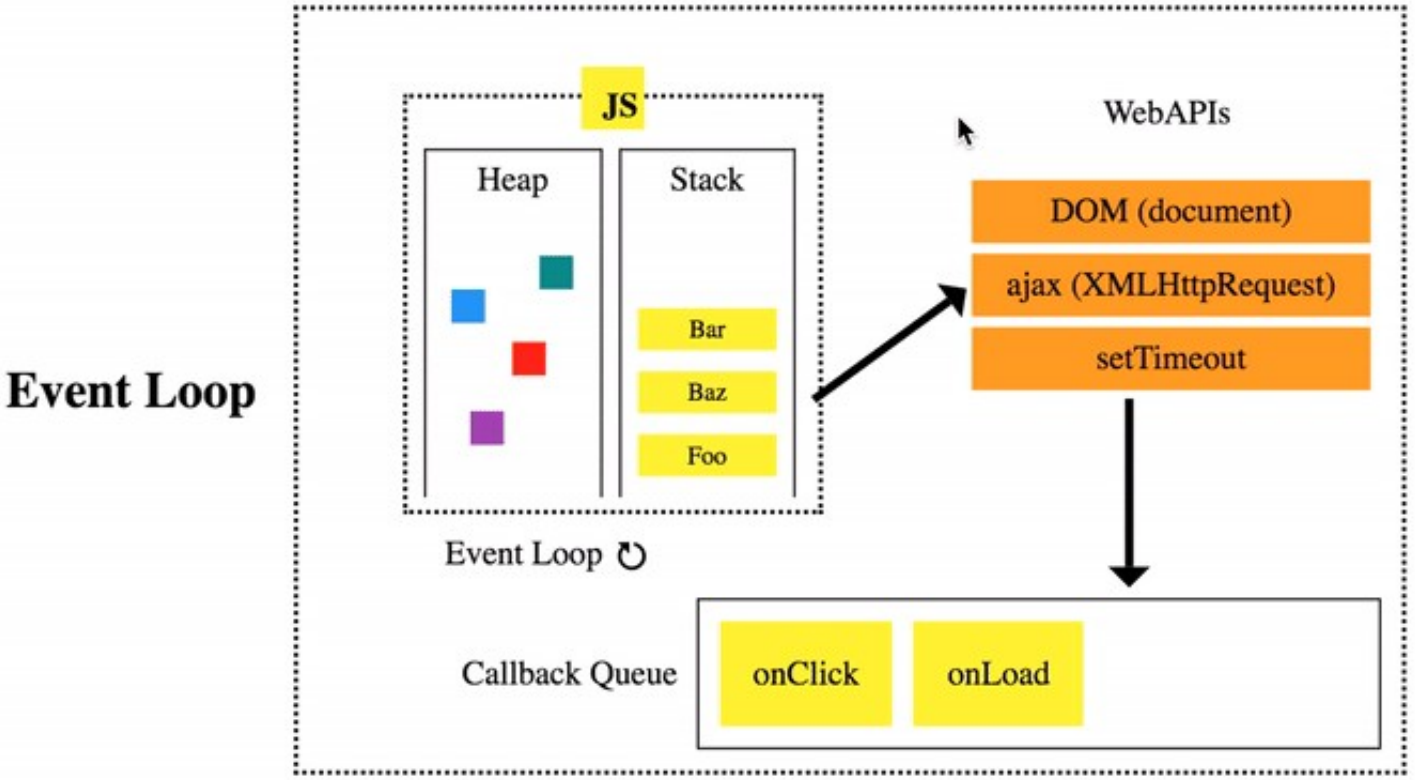
```
setTimeout(function() {
    console.log("Thong bao nay se duoc hien thi sau 3 giay !!!");
}, 3000);
```

Các dạng hàm callback khác

- Sử dụng hàm vô danh dưới dạng **hàm mũi tên**
  - Ví dụ:

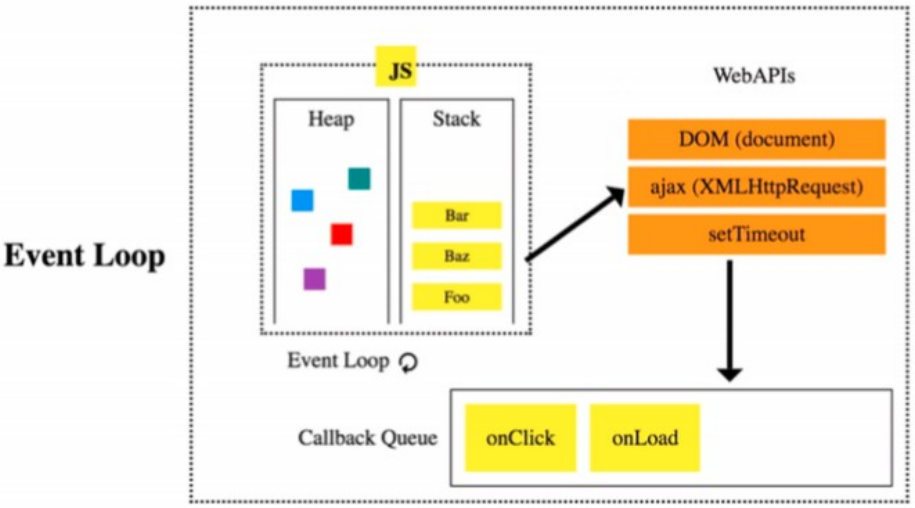
```
let value = 5;

setTimeout(() => {
    alert("Hien thi thong bao sau “ + value + “ giay !!!");
}, 5000);
```

Vòng lặp sự kiện trong Javascript



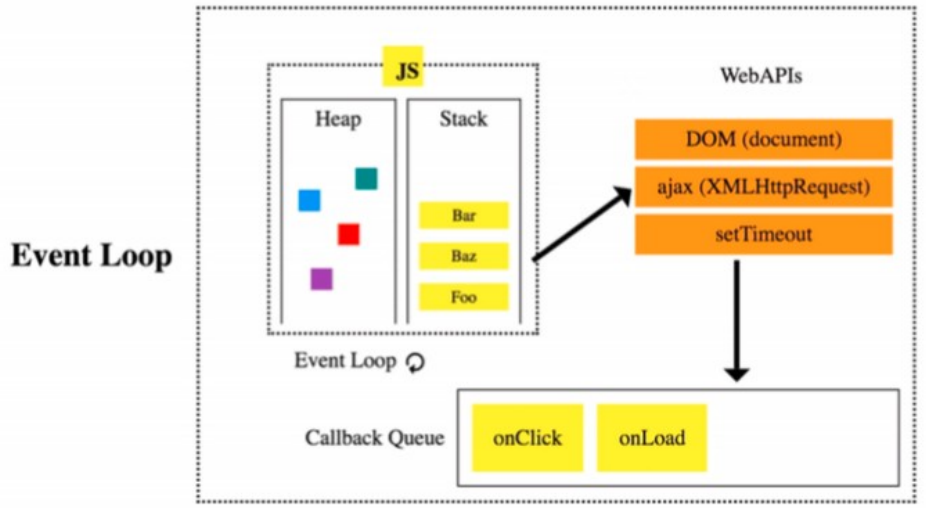
Vòng lặp sự kiện trong Javascript

- **Stack**
  - Đây là nơi mã nguồn Javascript **lần lượt** được đẩy vào bởi trình thông dịch **để thực thi** và sẽ được lấy ra khi việc thực thi hoàn thành.
  - Nếu gặp các **lệnh không đồng bộ** thì mã lệnh đó sẽ được **di chuyển đến hàng đợi callback** (hàng đợi sự kiện)



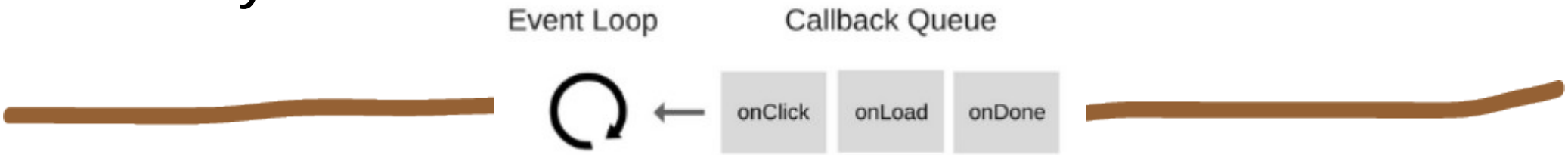
Vòng lặp sự kiện trong Javascript

- **Heap**
  - Đây là **vùng nhớ chứa các biến** bên trong chương trình.
- **Callback Queue**
  - Đây là nơi chứa các đoạn mã không đồng bộ đợi đến lượt để thực thi.
- **Event Loop**
  - Đây là một **vòng lặp kiểm tra liên tục Stack** xem có lệnh nào thực thi hay không.
    - Nếu không còn thấy lệnh nào trong Stack, thì hệ thống sẽ **kiểm tra Callback Queue;** Nếu trong Callback Queue có chứa mã thực thi thì mã đó sẽ được **chuyển sang Stack để thực hiện**.



Cơ bản về xử lý sự kiện trong Javascript

- **Lập trình hướng sự kiện** (Event-driven programming) là mô hình trong đó luồng chương trình được **xác định bởi các hành động** của người sử dụng như nhấp chuột, nhấn phím, ... Các hành động này được coi là các lệnh không đồng bộ.
- Khi một hành động xuất hiện, hệ thống sẽ tự động **đẩy hàm xử lý sự kiện** (đã được đăng ký) vào trong Callback Queue để chờ đến lượt xử lý.



Cơ bản về xử lý sự kiện trong Javascript

- Để xử lý sự kiện trong Javascript, ta cần hiểu 2 khái niệm sau
  - **Event handlers** là các **hàm do người lập trình tạo ra**, hàm này được gọi khi một sự kiện nào đó được kích hoạt.
    - Ví dụ: Hàm được gọi khi click chuột, hoặc nhấn một phím trên bàn phím, hoặc thay đổi kích thước cửa sổ, …
  - Mỗi event handler sẽ được gắn với một **event listener** là thủ tục dùng để **"nghe" (đợi) sự kiện xuất hiện**.
    - Mỗi event listener **cần được đăng ký trước** để nắm bắt sự kiện.
    - Khi sự kiện xuất hiện, event handler gắn với event listener sẽ được đẩy vào callback queue để chờ thực thi.

Các phương pháp nắm bắt sự kiện

- **Phương pháp 1**: Gán trực tiếp event handler với thuộc tính của đối tượng điều khiển
  - Không nên sử dụng cách này khi lập trình do khó quản lý code.
  - Code này trộn lẫn HTML và mã Javascript nên sẽ gây khó khăn trong việc phân tích và sửa đổi chương trình.

```
<!DOCTYPE html>
<html>

<head>
    <script type="text/javascript">
        function show_warning() {
            alert("Day la canh bao !!!");
        }
    </script>
</head>
<body>
    <button onclick="show_warning()">Show</button>
</body>

</html>
```

Các phương pháp nắm bắt sự kiện

- **Phương pháp 2**: Gán event handler cho thuộc tính của đối tượng.
  - Phương pháp này đã tách riêng được phần code lập trình và mã HTML.

```
<!DOCTYPE html>
<html>

<head>
    <title>Show warning</title>
</head>

<body>
    <button id="show-warning">Show</button>
</body>

<script type="text/javascript">

    let show_warning = function () {
        alert("Day la canh bao !!!");
    }

    let button = document.getElementById("show-warning");

    button.onclick = show_warning;

</script>

</html>
```

Các phương pháp nắm bắt sự kiện

- **Phương pháp 3**: Đối với mỗi sự kiện, sử dụng **hàm addEventListener()** để đăng ký event listener, đồng thời gắn kèm event handler.
  - Khi sự kiện được kích hoạt, event handler gắn kèm với event listener sẽ được đưa vào Callback queue.

```
<body>
    <button id="show-warning">Show</button>
</body>

<script type="text/javascript">

    let show_warning = function () {
        alert("Day la canh bao !!!");
    }

    let button = document.getElementById("show-warning");

    button.addEventListener("click", show_warning);

</script>
```

Các phương pháp nắm bắt sự kiện

- **Phương pháp 3**:
  - Ta có thể gắn kèm nhiều event handler vào một event listener. Khi sự kiện được kích hoạt, tất cả event handler sẽ cùng được sử dụng.

```
<body>
    <button id="show-warning">Show</button>
</body>

<script type="text/javascript">

    let show_warning = function () {
        alert("Day la canh bao !!!");
    }

    let show_comment = function () {
        alert("Day la chu thich !!!");
    }

    let button = document.getElementById("show-warning");

    button.addEventListener("click", show_warning);
    button.addEventListener("click", show_comment);

</script>
```

Các phương pháp nắm bắt sự kiện

- **Phương pháp 3**:
  - Sử dụng hàm **removeEventListener()** để xóa event handler của một event listener.

```
<body>
    <button id="show-warning">Show</button>
</body>

<script type="text/javascript">

    let show_warning = function () {
        alert("Day la canh bao !!!");
    }

    let show_comment = function () {
        alert("Day la chu thich !!!");
    }

    let button = document.getElementById("show-warning");

    button.addEventListener("click", show_warning);
    button.addEventListener("click", show_comment);

    button.removeEventListener("click", show_comment);

</script>
```

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với **đối tượng Button**
  - Tạo Button trong file HTML

```
<button id=”idButton”>Thông tin hiển thị</button>
```

    - Sử dụng tag có tên là **button**
    - id của button là **idButton** được sử dụng để nhận diện Button
    - Dòng chữ **Thông tin hiển thị** sẽ được hiển thị bên trên Button

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với đối tượng Button
  - Xử lý **sự kiện click chuột** của Button
    - Cách 1:

```
document.getElementById("idButton").onclick = function () {
        .....
};
```

    - Cách 2:

```
var myButton = document.getElementById("idButton");

myButton.addEventListener("click", function() {
        .....
});
```

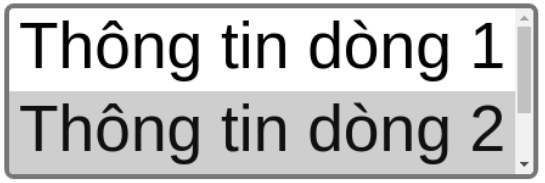
Ví dụ 1

- Viết chương trình cho chữ T xoay liên tục quanh gốc tọa độ. Sử dụng 2 Button làm nhiệm vụ:
  - Điều khiển chữ T xoay theo chiều kim đồng hồ.
  - Điều khiển chữ T xoay ngược chiều kim đồng hồ.
- ***Chú ý:***
  - Phương thức **window.requestAnimationFrame(func)** báo cho trình duyệt biết chương trình thực hiện ảnh động và yêu cầu trình duyệt gọi **hàm callback func** để cập nhật thông tin hình ảnh trước khi vẽ lại.

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với **đối tượng Select**
  - Tạo đối tượng Select trong file HTML

```
<select id=”idSelect” size=”2”>
    <option value=”0”>Thông tin dòng 1</option>
    <option value=”1”>Thông tin dòng 2</option>
    <option value=”2”>Thông tin dòng 3</option>
</select>
```



    - Sử dụng tag có tên là **select**
    - Đối tượng Select sẽ chỉ hiển thị trực tiếp 2 dòng do size=2
      - Nếu để **size=1** hoặc **không thiết lập thuộc tính size** thì đối tượng select sẽ chuyển sang dạng **đối tượng combobox**

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với đối tượng Select
  - Xử lý **sự kiện click chuột** của Select

```
var mySelect = document.getElementById("idSelect");

mySelect.addEventListener("click", function() {
    switch(mySelect.selectedIndex) {    // Lấy chỉ số dòng
        case ...
    }

    // Lấy giá trị của chỉ số dòng
    switch(mySelect.options[mySelect.selectedIndex].value) {
        case ...
    }
});
```

Ví dụ 2

- Dựa trên Ví dụ 1, thêm một hộp thoại lựa chọn để điều khiển tốc độ quay của chữ T. Giả sử ta có 5 tốc độ:
  - Quay siêu nhanh
  - Quay nhanh
  - Quay vừa
  - Quay chậm
  - Quay siêu chậm
- ***Chú ý:*** Thay đổi tốc độ quay bằng cách tăng, giảm góc quay tại mỗi lần vẽ.

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với bàn phím
  - Xử lý **sự kiện keydown** của bàn phím

```
document.addEventListener("keydown", function(event) {
    switch(event.key) {
        case "ArrowLeft":   // Phím mũi tên bên trái
            .....
        case "1":   // Phím số 1
            .....
        case "a":   // Phím chữ a (cả hoa và thường)
            .....
    }
});
```

Ví dụ 3

- Dựa trên Ví dụ 2, điều khiển hướng quay và tốc độ của chữ T dựa trên bàn phím. Giả sử ta định nghĩa các phím như sau:
  - Phím mũi tên trái và phải trên bàn phím điều khiển việc quay trái và phải.
  - Phím mũi tên lên và xuống trên bàn phím điều khiển việc tốc độ quay.

- ***Chú ý***: Sử dụng **hàm console.log** để in thông tin của các phím chức năng khác vào cửa sổ console của browser mỗi khi được nhấn.

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với **đối tượng Slider**
  - Tạo đối tượng Slider trong file HTML

```
<input id="idSlider" type="range" min="0" max="100" step="10" value="50" />
```

    - Sử dụng tag có tên là **input** cùng với thuộc tính **type=”range”**
    - Thuộc tính **min** và **max** thể hiện **khoảng giá trị** của Slider
    - Thuộc tính **step** là **giá trị mỗi bước di chuyển** của Slider
    - Thuộc tính **value** là **giá trị ban đầu** của Slider

Làm việc với sự kiện

- Xử lý sự kiện với đối tượng Slider
  - Xử lý **sự kiện thay đổi giá trị** của Slider

```
var mySlider = document.getElementById(“idSlider”);

mySlider.addEventListener(“change”, function() {
    var value = parseFloat(event.target.value);
});
```

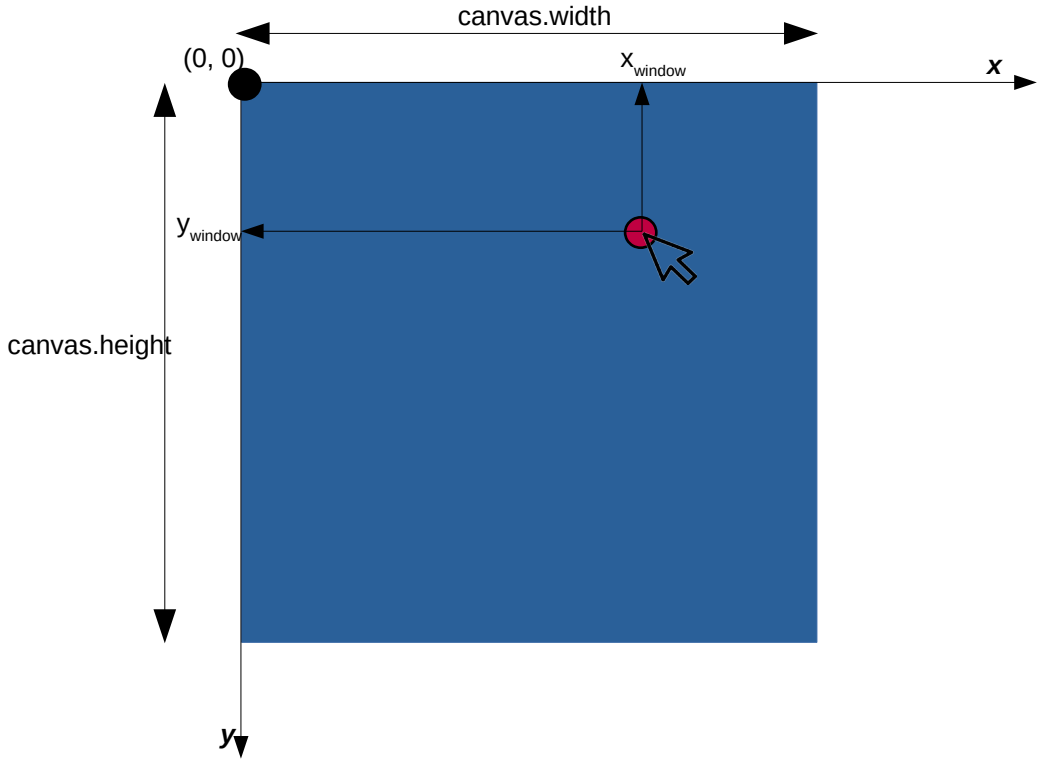
Ví dụ 4

- Dựa trên Ví dụ 3, thêm vào chương trình một Slider làm nhiệm vụ phóng to, thu nhỏ chữ T trong khi đang quay.

- ***Chú ý***: Giá trị trong khoảng (0, 1) là thu nhỏ chữ T.

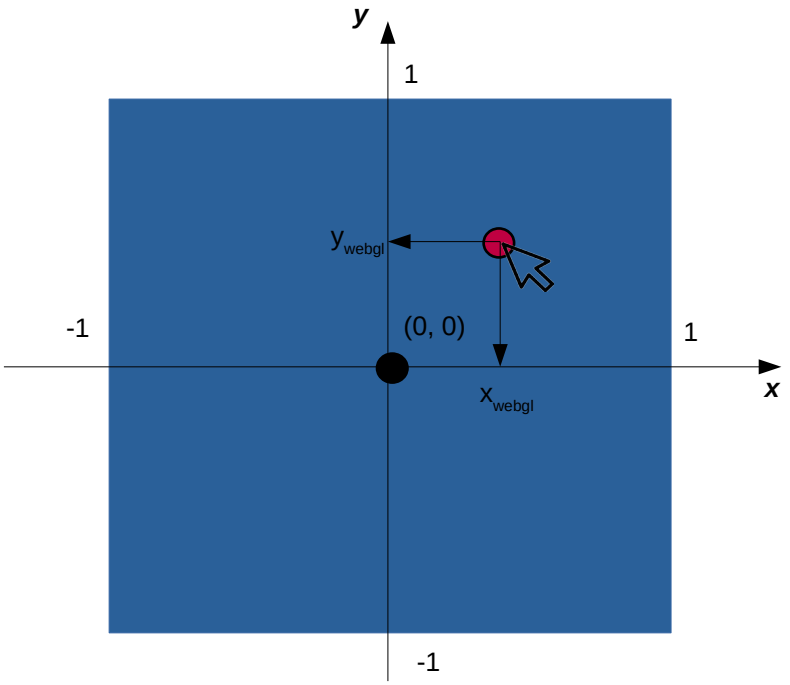
Sự kiện của các đối tượng cơ bản

- Trục tọa độ cửa sổ
  - Tọa độ con trỏ chuột trên trục tọa độ cửa sổ



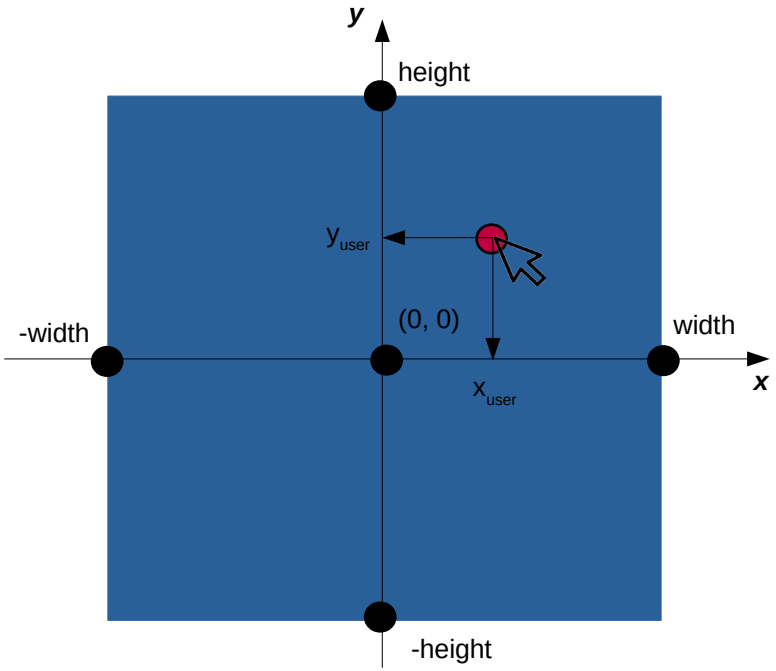
Biến đổi tọa độ con trỏ chuột

- Trục tọa độ của WebGL
  - Tọa độ con trỏ chuột trên trục tọa độ của WebGL



Biến đổi tọa độ con trỏ chuột

- Trục tọa độ của người sử dụng
  - Tọa độ con trỏ chuột trên trục tọa độ của người sử dụng



Biến đổi tọa độ con trỏ chuột

- Ta cần **biến đổi tọa độ con trỏ chuộ**t
  - Từ trục tọa độ cửa sổ sang trục tọa độ WebGL.
  - Từ trục tọa độ cửa sổ sang trục tọa độ của người sử dụng.
- Ta có **công thức biến đổi tọa độ từ trục tọa độ cửa sổ sang trục tọa độ WebGL** như sau:

$$x_{webgl} = -1 + \frac{2 * x_{window}}{width_{canvas}}$$

$$y_{webgl} = -1 + \frac{2 * (height_{canvas} - y_{window})}{height_{canvas}}$$

Biến đổi tọa độ con trỏ chuột

- Ta có **công thức biến đổi tọa độ từ trục tọa độ người sử dụng sang trục tọa độ WebGL** như sau:

$$x_{webgl} = \frac{x_{user}}{width_{canvas}}$$

$$y_{webgl} = \frac{y_{user}}{height_{canvas}}$$

Biến đổi tọa độ con trỏ chuột

- **Để biến đổi tọa độ từ trục tọa độ cửa sổ sang trục tọa độ người sử dụng** ta cần rút gọn công thức được lấy từ 2 công thức trên như sau:

$$\frac{x_{user}}{width_{canvas}} = -1 + \frac{2 * x_{window}}{width_{canvas}}$$

$$\frac{y_{user}}{height_{canvas}} = -1 + \frac{2 * (height_{canvas} - y_{window})}{height_{canvas}}$$

Biến đổi tọa độ con trỏ chuột

- Sau khi rút gọn ta có công thức biến đổi tọa độ **từ trục tọa độ cửa sổ sang trục tọa độ người sử dụng** như sau:

$$x_{user} = 2 * x_{window} - width_{canvas}$$

$$y_{user} = height_{canvas} - 2 * y_{window}$$

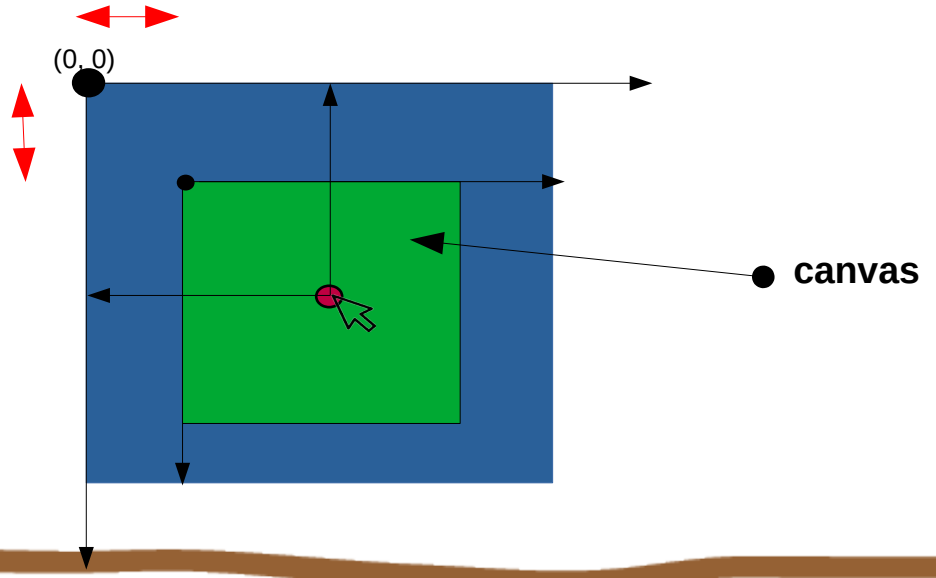
Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với chuột
  - Lấy vị trí click chuột **trên trục tọa độ cửa sổ**

```
canvas.addEventListener("click", function(event) {
    var x = event.clientX;  // Tọa độ x của điểm click
    var y = event.clientY;  // Tọa độ y của điểm click
});
```

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với chuột
  - Do **vùng hiển thị của canvas nằm bên trong nội dung của web browser**, do đó để xác định chính xác vị trí click chuột bên trong canvas, ta **cần phải hiệu chỉnh lại tọa độ** bằng cách trừ đi khoảng cách giữa cạnh canvas và mép nội dung của browser.



Ví dụ 5

- Viết chương trình mỗi khi click chuột lên màn hình, chương trình sẽ hiển thị một điểm ảnh màu đỏ tại vị trí vừa click chuột.

Sự kiện của các đối tượng cơ bản

- Xử lý sự kiện với cửa sổ
  - Xử lý **sự kiện resize** (thay đổi kích thước) của cửa sổ

```
window.addEventListener(“resize”, function() {
    var newWidth = innerWidth;   // Độ rộng mới của cửa sổ
    var newHeight = innerHeight;  // Chiều cao mới của cửa sổ
});
```

Ví dụ 6

- Dựa trên ví dụ 4, viết chương trình xử lý sự kiện resize cửa sổ sao cho mỗi lần thực hiện resize, viewport tự động điều chỉnh chiều rộng và chiều cao của vùng vẽ bằng ½ chiều cao của cửa sổ.
- ***Chú ý***: So sánh với các ví dụ cũ, không có sự kiện resize.

Hết Tuần 4

Cảm ơn các bạn đã chú ý lắng nghe !!!