CSCI-UA-4-005

Intro to Web Design + Computer Principles

Javascript: Day 3

Professor Emily Zhao

M/W 12:30PM – 1:45PM

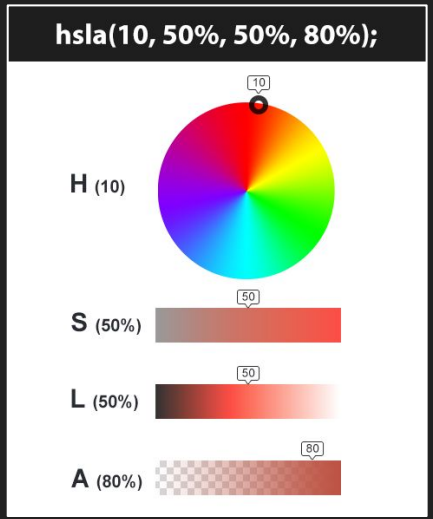
Agenda

- Warm-up: Flip a Coin (cont'd)
- Lecture:
  - Functions
  - DOM Events
- Finish Coin Flip Page
  - Add a button + Event handling
- Peer Programming: Airport Site
- CSS Transitions
- Demo:
  - To-Do List
  - Self-Destroying Website

Warmup: Randomly Change BG Color

On our Flip a Coin website, write Javascript to randomly change the background:

1. Use a DOM query to select the <body> element in the HTML document and store as a variable
2. Generate a random hue value between 0 and 359 using the HSL color model
3. Set the background color of the <body> element to a random color with full saturation and 50% lightness



Functions

Functions

Functions are blocks of reusable code that can be defined and invoked (called) to perform a specific task.

Functions play a crucial role in:

- Organizing and structuring code
- Promoting code reuse
- Enabling modular and maintainable programs

```
// Defining a function without parameters
function greet() {
  console.log('Hello, world!');
}

// Calling the greet function
greet();
```

```
// Defining a function with parameters
function addNumbers(a, b) {
  return a + b;
}

// Calling the addNumbers function with arguments 3 and 5
const result = addNumbers(3, 5);
console.log(result); // Output: 8
```

→ Let’s make `coinFlip()` and `randomBgColor()` functions

DOM Events

DOM Events

As you navigate the web, your browser registers different types of events.

Common events include:

- Clicking or tapping on a link
- Hovering or swiping over an element
- Resizing the browser window
- A web page loading

JavaScript can be used to respond to the multitude of events that occur within the DOM.

Keyboard Events

- keydown
- keyup
- keypress

Mouse Events

- click
- dblclick
- mousedown
- mouseup
- mouseover
- mouseout

Focus Events

- focus
- blur

Touch Events

- touchstart
- touchmove
- touchend
- touchcancel

Pointer Events

- pointerover
- pointerenter
- pointerdown
- pointermove
- pointerup
- pointercancel
- pointerout
- pointerleave
- gotpointercapture
- lostpointercapture

User Interface (UI) Events

- load
- unload
- error
- resize
- scroll

Mutation Events

- DOMSubtreeModified
- DOMNodeInserted
- DOMNodeRemoved
- DOMNodeInsertedIntoDocument
- DOMNodeRemovedFromDocument

Form Events

- input
- change
- submit
- reset
- cut
- copy
- paste
- select

Binding

Specifying which event will trigger the response is also known as “binding”.

There are three different ways to bind an event to an element:

— HTML event handler
— DOM event handler
— DOM Event listener *

* *preferred method*

HTML Event Handler

```
<button onclick="myFunction()">Click me</button>
```

***Downsides**: Mixing HTML markup with JavaScript can make the code less maintainable and harder to debug. It's generally considered a best practice to separate HTML and JavaScript code.*

DOM Event Handler

```
let btn = document.querySelector("button");
btn.onclick = myFunctionName;
```

***Downsides:** Assigning multiple event handlers to the same event on the same element will overwrite the previous assignment.*

DOM Event Listener

```
let btn = document.querySelector('button');
btn.addEventListener('click', myFunctionName);
```

***Most recommended!** They're most flexible and powerful. They allow you to attach multiple event handlers to a single event on a DOM element without overwriting existing ones.*

Event Handling

1. Select an element for the script to respond to
2. Specify which event will trigger the response
3. Run code specific to that event

```
// Step 1: Select an element for the script to respond to
const button = document.getElementById('myButton');

// Step 2: Specify which event will trigger the response
button.addEventListener('click', myFunction);

// Step 3: Run code specific to that event
function myFunction() {
  alert('Button clicked! This is the response to the click event.');
})
```

Flip a Coin (cont'd)

Now, we're going to add a button to our HTML.

Instead of flipping the coin and generating a random background color on reload, we are now going to bind those responses to a button click.

I'll show you all three ways to do it:

1. HTML event handler
2. DOM event handler
3. DOM event listener

Peer Programming: Airport Site

Without changing or adding any HTML, write Javascript to change what main image is shown (and be sure to change the srcset as well) depending on which colored nav image is selected:

1. Blue → LGA
2. Green → CDG
3. Red → NRT

Hints: There are query selectors that select one item and others that select multiple and return a list. Which one will you need to use?

CSS Transitions

CSS Transitions

CSS transitions are a way to smoothly animate the changes in CSS properties over a specified duration.

They provide a more elegant and visually appealing way to transition from one style to another.

Transitions can be applied to various CSS properties, such as colors, sizes, positions, and more.

```css
.element {
  background-color: "blue";
  transition-property: background-color; /* Specifies the property to transition */
  transition-duration: 0.5s; /* Specifies the duration of the transition */
  transition-timing-function: ease-in-out; /* Specifies the timing function */
  transition-delay: 0s; /* Specifies the delay before the transition starts */

  /* Or written all in one line */
  transition: background-color 0.5s ease-in-out 0s;
}

.element:hover {
  background-color: "red";
}
```

`this`

In event handlers, such as those used in HTML or with the DOM, **`this`** typically refers to the DOM element that triggered the event.

```
<button onclick="console.log(this)">Click me</button>
```

[To-Do List Demo]

Homework

- Happy Thanksgiving!
- Assignment #8 (due next Monday)