CSCI-UA-4-005

Intro to Web Design + Computer Principles

Javascript: Day 1

Professor Emily Zhao

HTML

Add and orders elements on a webpage. Like the *skeleton* of a webpage.

CSS

Handles the styling of your website. For example, fonts, colors, sizing, etc.

JavaScript

Adds action and allows user interactions. For example, buttons and text input fields.

			
Language	**HTML**	**CSS**	**Javascript**
Purpose	Structure, Objects, Things	Looks, Style	Actions
Syntax	<p> <h1> 	P {color: red;}	var x = 5;
Grammar	nouns	adjectives	verbs
Building	Walls, structure	Paint, curtains	Electrical, Plumbing, AC

Background

- JavaScript was was invented by Brendan Eich and introduced by Netscape in 1995.
- At that time, the Java language was ascendant and the name “JavaScript” was an attempt to ride this popularity.
- Eventually, browsers other than Netscape began to support JavaScript functionality, calling it “ECMAScript.”
- Today, JavaScript is not only a *lingua franca* of the web but a basis for many other computational media projects

Application

- As with CSS, JavaScript targets HTML elements to do something with them.
- There are three ways you can apply JavaScript to HTML:
  - Inline JavaScript
  - Embedded JavaScript
  - External JavaScript
- External and embedded JavaScript are preferable for their separation of content and behavior.

A Front-End Language

- Like HTML and CSS, JavaScript is usually rendered in the web browser.
- Because it’s rendered in the browser rather than on a server, JavaScript is considered a “front-end” language.
- A browser’s “JavaScript engine” interprets and executes JavaScript code in the browser.
- There are different JavaScript engines for different browsers.

Compatibility

- Computationally speaking, there isn't much JavaScript can't do; it's a robust programming language.
- Core functionality includes modifying HTML and CSS, communicating with the server, and storing data.
- We will use JavaScript to modify page content and style.
- As with any technology, it's good to consider when to use it and when it's unnecessary.

A **markup language**, like HTML, is designed to...

- Structure and present content
- Provide annotations to define elements such as paragraphs, headings, and links
- Its primary role is to describe how content should be displayed.

A **programming language**, like Javascript, is intended to...

- Write instructions that a computer can execute.
- Create algorithms, implement logic, and perform computations.
- With Javascript, you can access and manipulate elements on a page, respond to user actions, and perform calculations.

Today's Attendance
(via PollEverywhere)

pollev.com/emilyzhao

→ Have you taken Intro to Programming?

[Greeting Demo]

```
/* Greets a person based on the time of day */

let today = new Date();
let hour = today.getHours();



let greeting;
let paragraph = document.getElementById('greeting');

if (hour >= 18) {
  greeting = 'Good evening!';
} else if (hour >= 12) {
  greeting = 'Good afternoon!';
} else if (hour >= 0) {
  greeting = 'Good morning!';
} else {
  greeting = 'Hello!';
}

paragraph.textContent = greeting;
```

Javascript can access current **date and time** to personalize content

Variables are used to store data needed by the program

Conditional logic allows Javascript to make decisions and adapt the content

Javascript can **interact with HTML** to dynamically update the page content

Variables

Variables

“Buckets” that store information in your computer’s memory

```
let speed = 5;
```

```
let name = "Emily";
```

Naming Variables

- Can’t contain spaces (can use "_" in place) or special characters (!@#%^&*)
- Can only start with a letter or underscore ; can be followed by any alphanumeric character after that
- Can’t use Javascript’s "reserved" words
- Javascript is case-sensitive

Javascript Reserved Words

Words marked with* are new in ECMAScript 5 and 6

abstract	arguments	await*	boolean
break	byte	case	catch
char	class*	const	continue
debugger	default	delete	do
double	else	enum*	eval
export*	extends*	false	final
finally	float	for	function
goto	if	implements	import*
in	instanceof	int	interface
let*	long	native	new
null	package	private	protected
public	return	short	static
super*	switch	synchronized	this
throw	throws	transient	true
try	typeof	var	void
volatile	while	with	yield

Legal or Illegal variable name?

```
class = 2
```

```
class_avg = 70
```

```
classAvg = 99
```

```
_class_avg = 99
```

```
2ndclassavg = 88
```

```
classavg! = 99
```

Legal or Illegal variable name?

```
class = 2
```
❌ `class` is a reserved word

```
class_avg = 70
```
✅

```
classAvg = 99
```
✅

```
_class_avg = 99
```
✅

```
2ndclassavg = 88
```
❌ cannot start with number

```
classavg! = 99
```
❌ alphanumeric only

Common Variable Naming Conventions

```
rockettopspeed = 100;
```
→ valid, but hard to read

```
rocket_top_speed = 100;
```
→ underscored

```
rocketTopSpeed = 100;
```
→ camelCase

Data Types

JavaScript needs to know how to set aside memory in your computer based on what kind of information you want to store:

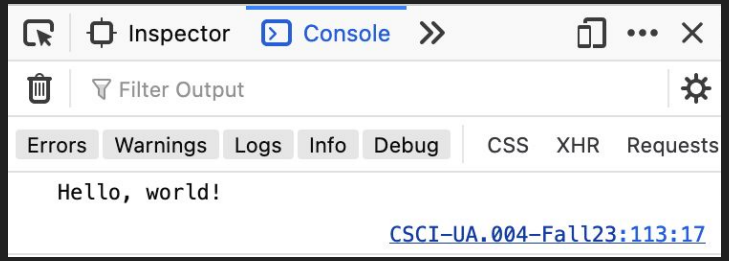
- Strings (character-based data)
- Numbers
- Boolean Values (true / false)
- Arrays (a list of values)
- null / undefined

Testing + Visualizing Our Data

Testing + Visualizing Our Data

`console.log()` allows you to print messages to the Javascript console in the developer tools.

```
console.log("Hello, world!");
```



`alert()` displays an alert box with message and an OK button. Only use this for special cases!

```
alert("Hello, world!");
```

Javascript Expressions + Operators

Math Operators

`+`	addition
`-`	subtraction
`*`	multiplication
`/`	division
`%`	modulus (remainder)
`++`	incremeter (add 1 to current value)
`--`	decremeter (subtract 1 to current value)

Relational Operators

`>`	greater than
`<`	less than
`==`	equality (compares values only)
`===`	strict equality (compares values and data type)
`<=`	greater than or equal to
`>=`	less than or equal to
`!=`	not equal to
`!==`	strictly not equal to

Logical Operators

&&	and
||	or
!	not

Boolean Expressions

A boolean expression is a logical statement that evaluates to either true or false. You can think of them as “yes or no” questions. They are formed using comparison/relational operators and logical operators.

```
let x = 5;
let y = 10;
let z = 3;

(z > 0)                 // true
(y > 20)                // false
(x * 2 == y)            // true
(x > z && x < y)        // true
(x >= 5 && z >=5)       // false
(z == 3 || x == 8)      // true
```

Conditionals

Conditionals are constructs that allow you to control the flow of your code based on specified conditions. They are used to make decisions and execute different code blocks depending on whether a given condition is true or false.

```
let x = 10;
if (x > 5) {
  console.log('x is greater than 5');
}
```

if-else

```
let y = 3;
if (y % 2 === 0) {
  console.log('y is even');
} else {
  console.log('y is odd');
}
```

if - else if - else

```
let grade = 75;
if (grade >= 90) {
  console.log('A');
} else if (grade >= 80) {
  console.log('B');
} else if (grade >= 70) {
  console.log('C');
} else {
  console.log('F');
}
```

Date Object

The Date object is a built-in object that provides methods for working with dates and times.

It represents the number of milliseconds since January 1st, 1970 (UTC).

```
Date.now()       // returns the current number of milliseconds
.getHours()      // returns the hour (military time) of the current local time
.getMinutes()    // returns the minutes of the current local time
.getSeconds()    // returns the seconds of the current local time
.getDay()        // returns the day of the week as a number (Sunday is 0)
```

[Class Today?]

Homework — Assignment #7 (due midnight)