

The Building Blocks Part 2

CS 130: Intro to programming with JavaScript

Announcements

- Project proposal due Sunday
- Tutorial 6: Make a photo gallery
- Forthcoming: sign up for a consulting slot!

Warm-Up: Drawing Activity

But first, one more built-in function...

```
element.insertAdjacentHTML(position, text);
```

Valid options for “**position**”:

1. 'afterbegin': Just inside the element, before its first child.
2. 'beforeend': Just inside the element, after its last child.
3. 'beforebegin': Before the element itself.
4. 'afterend': After the element itself.

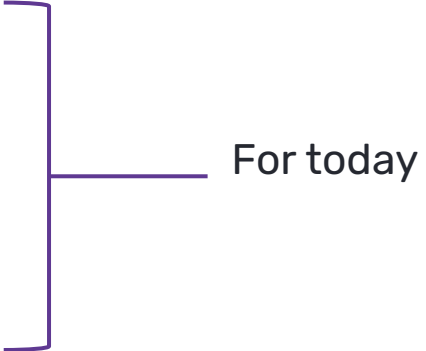
“**text**” can be any valid HTML string

Activity: Make Shapes

1. When the user clicks on the SVG element, draw a circle where the user clicked.
2. How do you make the circle honor the color in the “color” textbox?
3. How do you make the circle honor the size in the “size” textbox?
4. How do you make the circle honor the shape in the “shape” dropdown menu? (preview for next week)
5. How do you make a circle draw when the user mouses over the SVG element?
 - a. Drags?

Outline

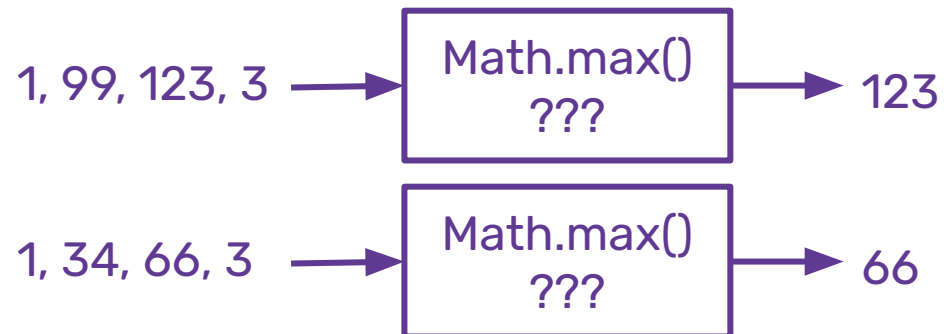
In this class, we're going to do a high-level overview of programming, and the fundamental constructs of JavaScript

1. **Data**
data types and variables
 2. **Statements & Expressions**
constants, operators, and functions
 3. **Events**
 4. **Control**
conditionals, iteration, and looping
- 
- For today

Functions

Functions: provide a way to encapsulate and reuse code

- Functions allow you to encapsulate and group lines of code.
- With functions, you can perform the same operations over and over using different data
 - Example from Wednesday: drawing a circle but changing the size, position, and color.
- JavaScript has many built-in functions
- You can also create your own functions



To run a function, you have to call or “invoke” it

In order to run a function, you have to invoke it and give it the data it needs, using the correct type:

```
> Math.max;                                // tells you that max is a function
```

```
> Math.max(2);
```

```
> Math.max(2, 55, 29, 88, 7);
```

```
88
```

Creating your own functions

Functions: header and body

```
const nameOfFunction = (parameters) => {
```

```
  statement 1;
```

```
  statement 2;
```

```
  ...
```

```
  return some_value;
```

```
};
```

HEADER

Specifies the name of the function and the data that it needs. Also called the **SIGNATURE**

BODY

One or more indented statements that the function will execute when called

Assign your **function** to a variable to name it (camel case) – use **const** keyword

parameters (the way we pass data to a function)

fat arrow

```
const nameOfFunction =  
  statement(s);  
  return some_value;  
};
```

end of statement block (end of header)

(parameters)

=>

{

start of statement block (end of header)

function body has 1 or more **statements**, which have same indentation level (usually 4 spaces)

An optional **return statement** to return a value from the function

Syntax: Creating your own functions

// ES6 Syntax:

```
const doSomething = (ev) => {  
    console.log(ev); //access to the event  
}
```

```
document.querySelector('button').onclick = doSomething;
```

There are alternative syntaxes for creating functions

// Alternative syntax

```
function addTwoNums(num1, num2) {  
    return num1 + num2;  
}
```

// Using arrow function (ES6 syntax):

```
const addTwoNums = (num1, num2) => {  
    return num1 + num2;  
}
```

Demo

Function Inputs: Parameters and Arguments

- Some functions don't accept any data/arguments
- Some functions require certain kinds of data/arguments
- Some functions allow you to pass in multiple optional data/arguments

Terminology: Parameters & Arguments

arguments: the data that you pass into a function

parameters: local variables, inside a function, that are assigned when the function is invoked

function definition: tells you which parameters are required and which are optional (if any)

Events

Events

When JavaScript is used in HTML pages, JavaScript can "react" to particular "events," which include (among others):

- onchange
- onclick
- onmouseover
- onmouseout
- onkeydown
- onload

Events

Events are comprised of two parts:

1. **Event Listeners:** refer to the particular interaction / thing to which you want to listen
2. **Event Handlers:** function (i.e. snippets of code) that you want to execute when the event listener triggers the event.

Example: Generic Event Handler

// event handler:

```
const sayHello = () => {  
    alert('Hello');  
};
```

// attach the event handler to the event listener:

// “when #my_button is clicked, invoke the sayHello function”

```
document.querySelector('#my_button').onclick = sayHello;
```

// alternative syntax

```
document.querySelector('#my_button').addEventListener('click', sayHello);
```

Demo: 02_events.js

Conditional Statements

Conditional Statements

Conditional statements are like a choose your own adventure novel:

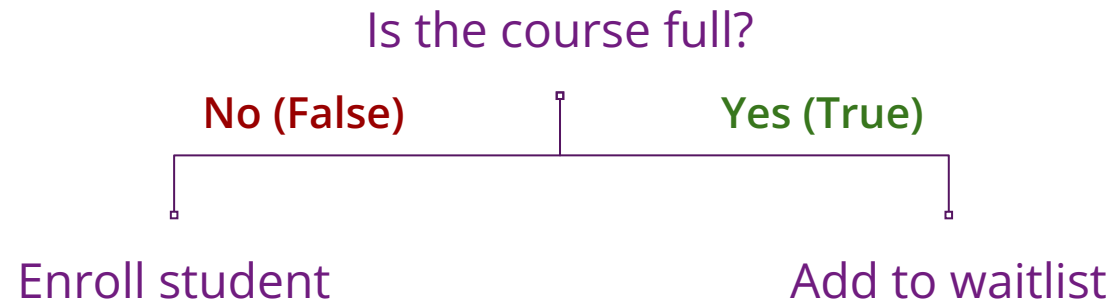
- If you choose to open Door #1, one thing happens, otherwise something else happens
- New choices can also be made as you step through new doors
- After a series of choices, many different outcomes become possible
- Each reader ends up in a different place: same book, different outcome

Computer programs work the same way. Depending on the data that's passed into the program, one part of your program will execute while another part gets skipped over

Conditional Statements

Conditional statements enable you to skip over some statements and execute others, depending on whether a condition is True or False.

EXAMPLE



If Statement

If the condition evaluates to True, the first block executes.

If the condition evaluates to False, the second block executes.

```
if (condition) {  
    statement 1;  
    statement 2;  
}  
else {  
    statement 1;  
    statement 2;  
}
```

CONDITION

Boolean expression that evaluates to True or False.

Comparison Operators

=	Assignment
==	Equality
===	Strict Equality (both values and data types match)
!=	Not Equal
>, >=	Greater than; greater than or equal to
<, <=	Less than; Less than or equal to

Before doing the color mixer demo, logical operators...

Logical Operators

Operator	Meaning	Explanation
&&	and	If both operands are true, then the “and expression” also evaluates to true. Otherwise, the “and expression” evaluates to false.
	or	If either or both of the operands are true, then the “or expression” also evaluates to true. Otherwise, the “or expression” evaluates to false.
!	not	if the operand is false, then the “not” of the operand is true (and vice versa).

Lists

Lists (AKA “Arrays”)

```
const myList = ["charlie", "freddie", "lucy"]
```

//access individual items:

```
console.log(myList[0]);
```

```
console.log(myList[1]);
```

```
console.log(myList[2]);
```

```
console.log(myList.length);    //get length of list
```

```
myList.push("jimena");        // append item
```

```
myList.pop();                 // remove item
```

0	"charlie"
1	"freddie"
2	"lucy"

Examples of Lists

1. A list of image urls (list of strings)
2. A list of names (list of strings)
3. A list of Tweets (list of objects)
4. A list of Songs (list of objects)

Code Example: List of Names

List of Strings

- Download lecture files and open the **02-list-of-strings** folder in VS Code
- How could you print all of the names?
- Preview: for...of loop

List of Images

- Download lecture files and open the **03-photos** folder in VS Code
- How can you output all of the images?

Objects

2. Objects: Representing Complex Entities

Some kinds of data (i.e. data about people, places, and belongings) are complex, and need more complex data structures to represent them in ways that a computer can understand and easily store and manipulate.

Often entities that are made up of sub-entities, which are related in some way. Examples:

- Tweets, Facebook posts, etc.
- Student data (like in Caesar)
- IRS data
- Human resources data

Example

```
const person = {  
  name: "Jane",  
  pic: "http://knight.gamebanana.com/img/ico/sprays/patrick_star_preview_2.png",  
  score: 300  
};
```

Code Example: Object Demo

- Download lecture files and open the **04-single-object** folder in VS Code
- Practice with dot notation

How would you represent multiple players?

3. Many People → List of Objects

```
const people = [  
  {  
    name: "Jane",  
    pic: "http://knight.gamebanana.com/img/ico/sprays/patrick_star_preview_2.png",  
    score: 300  
  },  
  {  
    name: "Brenda",  
    pic: "https://3.bp.blogspot.com/SpongeBobStock5-25-13.png",  
    score: 10  
  },  
  {  
    name: "Wanda",  
    pic: "https://photos.com/avatar.png",  
    score: 60  
  }  
];
```

Code Example: List of Objects

- Download lecture files and open the **05-list-of-objects** folder in VS Code
- Check it out.
- Try and output the second and third person's profile to the web browser.

Outline

1. Lists
2. Dictionaries
- 3. for...of loops**
4. Templates

Iteration and Looping

1. Iteration refers to a method of visiting each item of a list in order to do something with each item
2. Loops are a way to repeat lines of code over and over again

Example

Q: How do I do something to every element in the following list:

```
const names = ['charlie', 'freddie', 'lucy'];
```

A:

```
for (const name of names) {  
    console.log(name);  
}
```

For .. Of Syntax

BLOCK

If the statements
that will
repeat for each
item in the array

```
for (const item of array) {  
  // do something  
  // with item  
  ...  
}
```

For item in array
assigns each item in the
array to a variable
name of your choice

Practice Time

Download course files. We will do the following exercises together:

- 01-list-of-strings
- 02-photos
- 03-list-of-objects