



Maydm

Web Development

# Day 8: Even More JavaScript

Loops, Objects & The DOM

# Warm up!

---

# Today's Schedule

---

## Morning:

- JavaScript So Far...
- Loops
- Project: Todo List
- Objects
- Objects vs Arrays

## Afternoon:

- Application Programming Interfaces
- Project: Pokedex
- JavaScript with HTML
- The Document Object Model

# JavaScript So Far

---

- Comments
- Variables
- Data Types
- Operating on Variables
- Template Literals
- Booleans
- Functions
- If Statements
- Arrays

# Loops

---

<https://www.flocabulary.com/unit/coding-for-loops/>

# For Loops

---

When you want to run something for a specific number of times, use a **for** loop.

```
for (var i = 0; i < 10; i++) {  
    console.log("=^.^=");  
}
```

# Anatomy of a For Loop

Use for keyword to declare a for loop

The terms for the loop to run. In this case:

- variable **i** starts at 0
- loop runs while as **i is less than 10**
- **i increases** after each loop

```
for (var i = 0; i < 10; i++) {  
    console.log("=^.^=");  
}
```

Everything inside the opening & closing curly braces runs each time the loop runs



# Practice with For Loops

---

Work through the Day 8 “For Loops” exercises on JS Bin.

# For Loops

---

When might a for loop be useful?

# While Loops

---

Sometimes you don't know how many times you need a statement to execute. You need it to run while a certain condition is true. Use a while loop:

```
while (hungry === true) {  
    eatBurrito();  
    hungry = false;  
}
```

# Anatomy of a While Loop

Use while keyword to declare loop

The terms for the loop to run. In this case while the hungry variable is true.

```
while (hungry === true) {  
  eatBurrito();  
  hungry = false;  
}
```

Everything inside the curly braces runs each time the loop runs:

- eatBurrito() function
- hungry variable switches to false

# While Loops

---

Sometimes the condition for a while loop always uses a comparison operator, like greater than, less than, or equal to.

```
while (hungryPeople > 0) {  
    makeBurrito();  
    hungryPeople--;  
}
```

# Anatomy of a While Loop

Use while keyword  
to declare loop

The terms for the loop to run. In this case while  
the hungryPeople variable is greater than 0.

```
while (hungryPeople > 0) {  
    makeBurrito();  
    hungryPeople--;  
}
```

Everything inside the curly braces  
runs each time the loop runs:

- makeBurrito() function
- hungryPeople decreases by 1

# Practice with While Loops

---

Work through the Day 8 “While Loops” exercises on JS Bin.

# While Loops

---

When might a while loop be useful?



# Project: Todo List

---

Open the “Todo List” starter code and fork the pen.

Just as before, we'll read through the code together, starting with the HTML.

# Objects

---

Sometimes you need to store data that's related but isn't the same as a list. An **object** is a collection of properties.

Think about a dog. A dog has the following properties:

- Type of dog
- Age
- Color of fur
- Owner
- Known tricks

# Anatomy of an Object

Use of var to  
declare variable

The name of  
the object

Everything  
inside the  
curly braces is  
a property of  
the object.

```
var dog = {  
  type: 'Golden Retriever',  
  age: 5,  
  fur: 'yellow',  
  name: 'Rover',  
  tricks: ['fetch', 'play', 'roll over'],  
  vaccinated: true  
}
```

Each property is  
made up of a **key** and  
**value** pair.

The **key** in this case is  
'fur,' and the **value** is  
'yellow.'

# Storing Data in Objects

---

All data types can be stored as **values** inside an object. Notice we have 3 strings, a number, an array, and a Boolean stored as **values**.

```
var dog = {  
  type: 'Golden Retriever',  
  age: 5,  
  fur: 'yellow',  
  name: 'Rover',  
  tricks: ['fetch', 'play dead', 'roll over'],  
  vaccinated: true  
}
```

# Accessing the Data in Objects

---

Data (values) inside objects can be accessed using the **keys**. In this example, the keys include type, age, fur, owner, tricks, and vaccinated.

```
var dog = {  
  type: 'Golden Retriever',  
  age: 5,  
  fur: 'yellow',  
  name: 'Rover',  
  tricks: ['fetch', 'play dead', 'roll over'],  
  vaccinated: true  
}
```

# Accessing the Data in Objects

---

Values can be accessed using the **variable name**, a **period**, and a **key**. This is called **dot notation**.

```
var dog = {...}  
  
dog.type; // returns 'Golden Retriever'  
dog.age; // returns 5  
dog.tricks; // returns ['fetch', 'play dead', 'roll  
over']
```

# Accessing the Data in Objects

---

Values can also be accessed using **bracket notation**, using the **variable name, the key, and brackets**.

```
var dog = {...}

dog['type'];    // returns 'Golden Retriever'
dog['age'];     // returns 5
dog['tricks'];  // returns ['fetch', 'play dead',
                 'roll over']
```

# Accessing the Data in Objects

---

If you want to access an array that's stored inside an object, start with the variable and property, then add an index!

```
var dog = {...}

dog['tricks'];           // returns ['fetch', ...]
dog['tricks'][0];        // returns 'fetch'
dog.tricks[0];           // returns 'fetch'
```



# Adding Data to Objects

---

Add properties to an existing object by using dot or bracket notation by giving a **key** and setting it equal to the **value**.

```
var dog = {...}  
  
// These do the same thing.  
dog.owner = "Jane";  
dog['owner'] = "Jane";
```

# Overwriting Data in Objects

---

You can also overwrite a property's value by reassigning it using the same bracket or dot notation.

```
var dog = {...}  
  
dog.name = "Fido";  
dog['name'] = "Fido";
```

# Deleting Data in Objects

---

You can also delete a property from an object using the delete keyword and the key.

```
var dog = {...}  
  
delete dog.tricks;  
delete dog['tricks'];
```

# Practice with Objects

---

Work through the Day 8 “Objects” exercises on JS Bin.

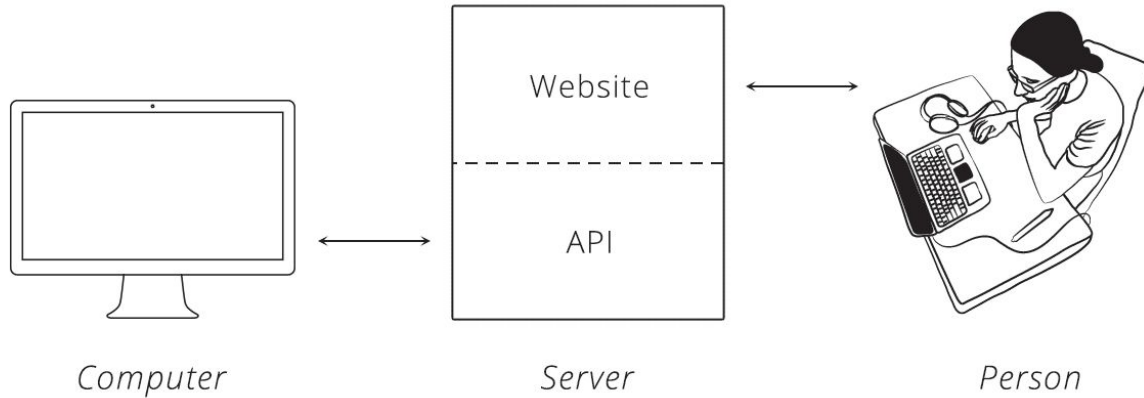
# Arrays vs Objects

---

When would you want an object instead of an array?

# APIs

**Application Programming Interfaces** are software that allows two programs to talk to one another. On the internet, APIs are what allow a server to return data to a user's web browser.



# JSON

---

Many web APIs return data in **JSON** format: JavaScript Object Notation. JSON looks like JavaScript but it is a separate language used to deliver data in a reliable format.

Javascript:

```
var student = {  
  firstName: "Joe",  
  lastName: "Smith"  
}
```

JSON:

```
{  
  "firstName": "Joe",  
  "lastName": "Smith"  
}
```

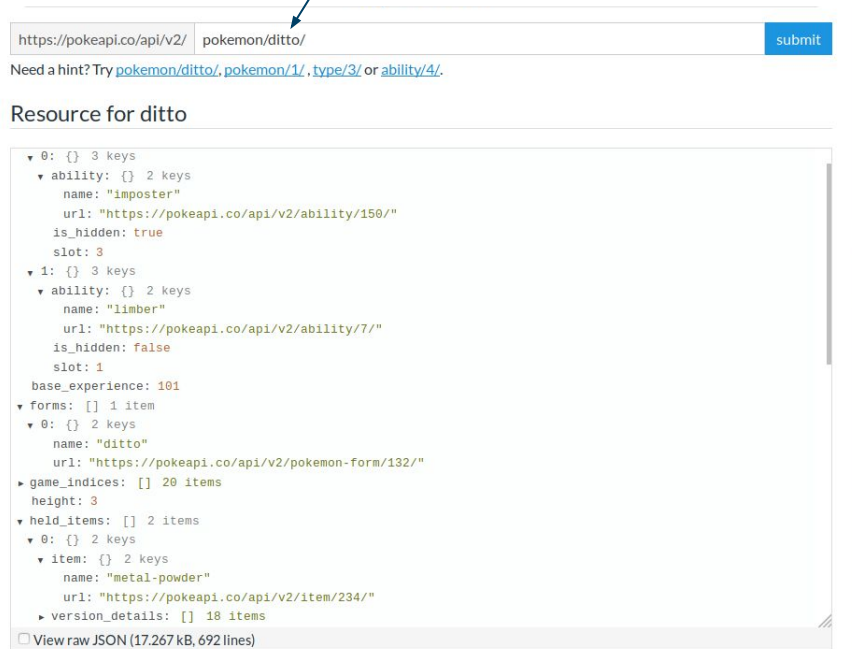
# Pokemon API

Check out how JSON and APIs work by playing with the Poke API.

Visit <https://pokeapi.co>

Replace “ditto” with your favorite Pokemon to see their data in JSON format

Replace ditto with your favorite pokemon



https://pokeapi.co/api/v2/ pokemon/ditto/ submit

Need a hint? Try [pokemon/ditto/](#), [pokemon/1/](#), [type/3/](#) or [ability/4/](#).

### Resource for ditto

```
{
  "abilities": [
    {
      "ability": {
        "name": "imposter",
        "url": "https://pokeapi.co/api/v2/ability/150/"
      },
      "is_hidden": true,
      "slot": 3
    },
    {
      "ability": {
        "name": "limber",
        "url": "https://pokeapi.co/api/v2/ability/7/"
      },
      "is_hidden": false,
      "slot": 1
    }
  ],
  "base_experience": 101,
  "forms": [
    {
      "form": {
        "name": "ditto",
        "url": "https://pokeapi.co/api/v2/pokemon-form/132/"
      }
    }
  ],
  "game_indices": [
    {
      "game_index": 1,
      "is_default": true
    },
    {
      "game_index": 2,
      "is_default": true
    },
    {
      "game_index": 3,
      "is_default": true
    },
    {
      "game_index": 4,
      "is_default": true
    },
    {
      "game_index": 5,
      "is_default": true
    },
    {
      "game_index": 6,
      "is_default": true
    },
    {
      "game_index": 7,
      "is_default": true
    },
    {
      "game_index": 8,
      "is_default": true
    },
    {
      "game_index": 9,
      "is_default": true
    },
    {
      "game_index": 10,
      "is_default": true
    },
    {
      "game_index": 11,
      "is_default": true
    },
    {
      "game_index": 12,
      "is_default": true
    },
    {
      "game_index": 13,
      "is_default": true
    },
    {
      "game_index": 14,
      "is_default": true
    },
    {
      "game_index": 15,
      "is_default": true
    },
    {
      "game_index": 16,
      "is_default": true
    },
    {
      "game_index": 17,
      "is_default": true
    },
    {
      "game_index": 18,
      "is_default": true
    },
    {
      "game_index": 19,
      "is_default": true
    },
    {
      "game_index": 20,
      "is_default": true
    }
  ],
  "height": 3,
  "held_items": [
    {
      "item": {
        "name": "metal-powder",
        "url": "https://pokeapi.co/api/v2/item/234/"
      },
      "version_details": [
        {
          "version": "Alpha",
          "rate": 5
        },
        {
          "version": "Beta",
          "rate": 5
        },
        {
          "version": "Gamma",
          "rate": 5
        },
        {
          "version": "Delta",
          "rate": 5
        },
        {
          "version": "Epsilon",
          "rate": 5
        },
        {
          "version": "Zeta",
          "rate": 5
        },
        {
          "version": "Eta",
          "rate": 5
        },
        {
          "version": "Theta",
          "rate": 5
        },
        {
          "version": "Iota",
          "rate": 5
        },
        {
          "version": "Kappa",
          "rate": 5
        },
        {
          "version": "Lambda",
          "rate": 5
        },
        {
          "version": "Mu",
          "rate": 5
        },
        {
          "version": "Nu",
          "rate": 5
        },
        {
          "version": "Xi",
          "rate": 5
        },
        {
          "version": "Omicron",
          "rate": 5
        },
        {
          "version": "Pi",
          "rate": 5
        },
        {
          "version": "Rho",
          "rate": 5
        },
        {
          "version": "Sigma",
          "rate": 5
        },
        {
          "version": "Tau",
          "rate": 5
        },
        {
          "version": "Upsilon",
          "rate": 5
        },
        {
          "version": "Phi",
          "rate": 5
        },
        {
          "version": "Chi",
          "rate": 5
        },
        {
          "version": "Psi",
          "rate": 5
        },
        {
          "version": "Omega",
          "rate": 5
        }
      ]
    }
  ],
  "name": "ditto",
  "order": 132,
  "species": {
    "name": "ditto",
    "url": "https://pokeapi.co/api/v2/species/132/"
  },
  "status": "available",
  "types": [
    {
      "type": {
        "name": "normal",
        "url": "https://pokeapi.co/api/v2/type/1/"
      }
    }
  ],
  "weight": 4.0
}
```

☐ View raw JSON (17.267 kB, 692 lines)



# Using JS with HTML

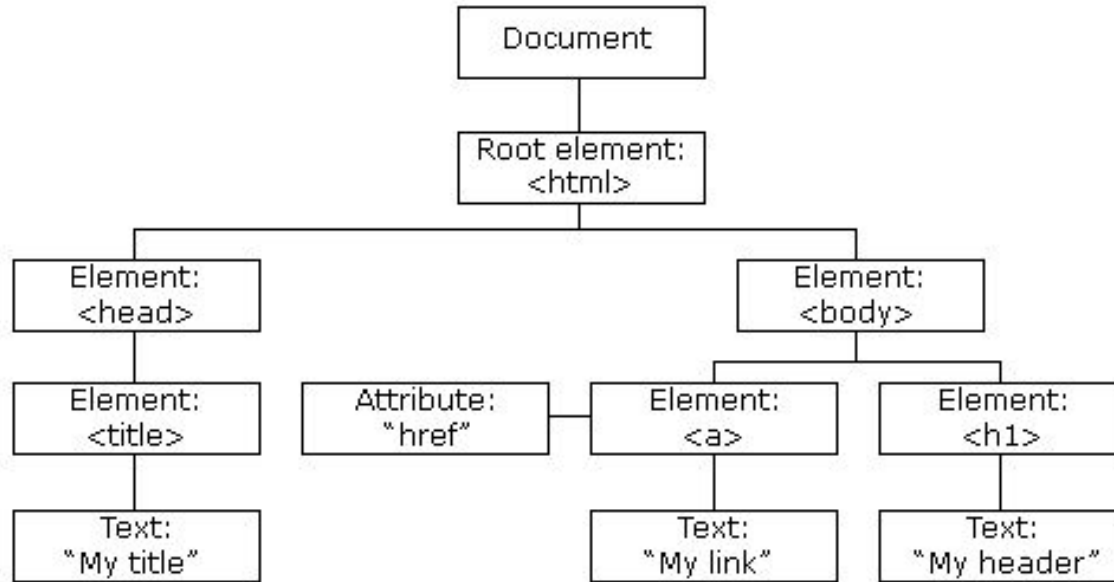
---

JavaScript works with HTML using the **Document Object Model**.

The DOM defines how a document (the website, in this case) is structured and accessed.

The DOM changes with each web page, based on the HTML of the page.

# The Document Object Model



# DOM Methods

---

You've already seen one DOM method regularly in the JS projects you've been working on:

```
document.getElementById(element);
```

Remember, methods are built-in functions that belong to specific objects and can only be used by those objects. This method can only be used by the **document** object.

# Commonly Used DOM Methods

---

- `document.getElementById(id)`
- `document.getElementsByTagName(name)`
- `document.createElement(name)`
- `parentNode.appendChild(node)`
- `element.innerHTML()`
- `element.setAttribute()`
- `element.getAttribute()`
- `element.addEventListener()`
- `element.insertAdjacentHTML()`
- `window.onload()`
- `window.scrollTo()`
- `console.log()`

# Selecting a DOM element by ID

---

You've seen the `getElementById` DOM method before in some of the projects. Because IDs are unique on a page, it is a very useful tool for manipulating HTML.

```
var testDiv = document.getElementById('test');  
testDiv.innerHTML = 'TESTING!'
```

# Selecting a DOM element by ID

---

The two chunks of code below do the same thing, though generally the second is considered better code. Why do you think that is?

```
document.getElementById('test').innerHTML = 'TESTING!'
```

// is the same as

```
var testDiv = document.getElementById('test');  
testDiv.innerHTML = 'TESTING!'
```

# Selecting a DOM element by HTML Tag

---

Another way to select elements is by HTML tag with the DOM method **getElementsByTagName**, which takes the tag as an argument. This returns an array of elements, which can be accessed through the indices.

```
var allDivs = document.getElementsByTagName("div");  
  
allDivs[0].innerHTML = 'This is the first div!';  
allDivs[1].innerHTML = 'This is the second div!';
```

# Selecting a DOM element by CSS Selector

---

Using the DOM method `querySelector`, you can find an element with any selector that works in CSS, including class, ID or tag. Selectors can also be combined to find specific tag types with classes or IDs. This returns the first element that matches the selector.

```
var example = document.querySelector('.example');  
// selects the first element with the class "example"  
  
var avengers = document.querySelector('ul#avengers');  
// selects the Unordered List with the ID "avengers"
```



# DOM Method: Changing Text

---

If you're only changing the text of an element, use the **innerText** method.

```
var test = document.getElementById('test');  
  
test.innerText = 'TESTING!';
```

# DOM Method: Changing Style

---

You can change the style of an element with style and the attribute you want to change. Note: If the attribute would normally use a hyphen, it uses camel case in JavaScript.

```
var test = document.getElementById('test');  
  
test.style.fontSize = '20px';
```

# DOM Method: HTML Attributes

---

HTML elements often have attributes, like the src attribute for image tags. Using the `getAttribute` and `setAttribute` methods, you can see if an attribute exists, what it is, or change it to something else.

```
var catPhoto = document.getElementById('cat');  
  
catPhoto.setAttribute('src',  
  'https://knowyourmeme.com/photos/406282-grumpy-cat');
```

# DOM Method: Adding elements

---

The DOM method `insertAdjacentHTML` adds HTML to a specific point in the DOM. The HTML can be added in the following places:

- `'beforebegin'`: Before the element itself.
- `'afterbegin'`: Just inside the element, before its first child.
- `'beforeend'`: Just inside the element, after its last child.
- `'afterend'`: After the element itself.

# DOM Method: Adding elements

---

```
var element = document.getElementById('someDiv');  
  
element.insertAdjacentHTML('beforebegin', '<div>This  
will be added before the selected element</div>');  
  
element.insertAdjacentHTML('beforeend', '<p>This will  
be added after the last child element of the div but  
before the end of the div</p>');
```

# Running JS on HTML Page

---

Using buttons is a common way to execute JS functions, and it is easy to implement using the “onclick” attribute. You can also add any necessary arguments to the function, just like you would using a JS file.

```
<button onclick="myFunction()">  
  Run a function  
</button>
```

# All Girls Gaming Hackathon

---

What would you like to create now that you know how to program?



# Reflection

---

Write in your journal about how you feel or what you learned today.

Prompts:

- How are objects and arrays similar? How are they different?
- APIs are used all over the web to share data. Can you think of a site you use that uses them? How are they used?
- Did you experience any frustration today while learning about JavaScript? How did you deal with it?