Python

```python
# Variables
full_name = "Jane Hacker"
pi = 3.14

# lists ("Arrays" in JS)
names = ["John", "Paul", "G"]

# dicts (similar to "Objects")
translation = {
    "ola": "Hello",
    "oi": "hi",
}

# For loops
for name in names:
    print("name:", name)

# While loops
x = 0
while x < 3:
    print("X:", x)
    x = x + 1


# If-statements
if full_name == "Jane":
    print("Hi, Jane!")
elif full_name == "Alice":
    print("Hey Alice")
else:
    print("Don't know you")


# Functions
def greeter(name):
    print("Hi", name)
greeter("Bob")


# Conjunctions
if age < 18 and drink == "beer":
    print("Too young kiddo")

if age > 18 or drink == "soda":
    print("Great choice")

# Class syntax
class User(BaseUser):
    def __init__(self, name):
        self.name = name
        self.logged_in = False
    def log_in(self):
        self.logged_in = True
user = User("janeqhacker")




# Making request (Synchronous)
response = requests.get("cnn.com")
data = response.json()
print("Resp:", data)
```

JavaScript

```javascript
// Variables
let fullName = "Jane Hacker";
const pi = 3.14;

// Arrays ("lists" in Py)
let names = ["John", "Paul", "G"];

// Objects (similar to "dicts")
let translation = {
    ola: "Hello",
    oi: "Hi",
};

// For loop
for (let name of names) {
    console.log("name:", name);
}
// While loops
let x = 0;
while (x < 3) {
    console.log("X:", x);
    x++;
}
// If-statements
if (fullName === "Jane") {
    console.log("Hi, Jane!");
} else if (fullName === "Alice") {
    console.log("Hey Alice");
} else {
    console.log("Don't know you");
}

// Functions
function greeter(name) {
    console.log("Hi", name);
}
greeter("Bob")

// Conjunctions
if (age < 18 && drink === "beer") {
    console.log("Too young kiddo");
}
if (age > 18 || drink === "soda") {
    console.log("Great choice");
}
// Class syntax
class User extends BaseUser {
    constructor(name) {
        this.name = name;
        this.loggedIn = false;
    }
    logIn() {
        this.loggedIn = true;
    }
}
let user = new User("janeqhacker");

// Making request (Asynchronous)
fetch("http://cnn.com")
.then(response => response.json())
.then(data => {
    console.log("Resp:", data);
});
```

Variable Declaration

let Declare a variable (block)
const Like `let`, cannot be reassigned.
var *Legacy* — similar to let, unscoped.

DOM Manipulation

```javascript
// Creating elements
let p = document
    .createElement("p");
p.textContent = "New Paragraph";

// Inserting elements into page
let d = document
    .querySelector("#some_id");
d.appendChild(p);

// Fetching many elements
let allImages = document
    .querySelectorAll("img");

// Add a class to all images
for (let img of allImages) {
    img.classList.add("Thumb");
}
```

Alternate function syntax

```javascript
let greeter = (name) => {
    console.log("Hi", name);
};
```

DOM terminology

DOM Initially generated from HTML, the Document Object Model is the current state of the page in the browser.
DOM traversal Finding elements in the DOM with JS
DOM manipulation Modifying the DOM with JS
event An interaction with the DOM (most common: `click`)

Asynchronous Terminology

asynchronous Instead of pausing ("blocking") for a slow operation, the asynchronous approach is to put-off starting the slow operation, then call a function ("callback") at a later time to signal it's done
callback A function passed as an argument to be called later when an event is triggered
promise Another popular way to do callbacks, with a `.then` syntax