**Next Gen JavaScript & TypeScript:**

1. **Const variables –** The value of avariables defined as a constant cannot be reassigned.
   1. **Example:**
      1. **const** username = “Max”;
2. **block scope** – A code segment surrounded by curly braces.
3. **var (global/function scope) –** The value of a variable defined with var can be reassigned.Variables defined outside of functions are available inside and outside of functions (global scope). Variables declared inside a function are only available within the function. Variables declared with var do not have block scope. Therefore, if a variable is declared within curly braces, it will be interpreted as a global variable, which can be updated unintentionally.
4. **let (block scope) –** Like var, the value of a variable defined with let can be reassigned.However, a variable or constant is only available with block scope (ex. if, for, while, functions, and curly braces), as well as any nested blocks. This allows us to avoid unintentional changes within the global scope.
5. **Arrow Function:** A syntax for writing a function as an expression. This is an essential feature of [functional programming](https://www.educative.io/answers/what-is-functional-programming?utm_campaign=brand_educative&utm_source=google&utm_medium=ppc&utm_content=performance_max&eid=5082902844932096&utm_term=&utm_campaign=%5BNew%5D+Performance+Max&utm_source=adwords&utm_medium=ppc&hsa_acc=5451446008&hsa_cam=18511913007&hsa_grp=&hsa_ad=&hsa_src=x&hsa_tgt=&hsa_kw=&hsa_mt=&hsa_net=adwords&hsa_ver=3&gclid=CjwKCAiA7vWcBhBUEiwAXieItod2x3_zZkl6yssk1ua9wsvwjdIN-OGAE7jPrFv1SaH5VK_ouiQetRoCqxcQAvD_BwE).
6. **Arrow Block Option 1:** 
   1. **Longhand:**
      1. const add = (a: number, b: number) => {return a + b};
   2. **Shorthand:** If the function’s return statement has only one expression, omit the return keyword and the curly braces.
      1. const add = (a: number, b: number) => a + b;
7. **Arrow Block Option 2:**
   1. **Longhand:**
      1. const printOutput = (output: string | number) => console.log(output);
   2. **Shorthand:** If the function has only one argument, you can omit the parenthesis, by declaring the variable with a function declaration .
      1. const printOut: (output: string | number) => void = output => console.log(output);
      2. button.addEventListener(‘click’, event => console.log(event));
   3. **Note: If the function has no parameters, you have to use an empty pair of parentheses:** 
      1. Syntax: () => {};
8. **Assigning Default Parameters to function arguments.**
   1. Note: Default parameters must be listed last, to maintain the parameter order.
      1. const add = (a: number, b: number = 1) => a + b;
      2. console.log(add(5));
9. **Spread Operator:** Tells JavaScript to pull out all elements in an array and add them as a list of individual values where you are using the operator. Use the spread operator whenever you need to obtain a comma separated list of array or object elements.
   1. **Copy an array**
      1. const hobbies = ['Sports', 'Cooking'];
      2. const activeHobbies = ['Hiking'];
      3. activeHobbies.push(...hobbies);
   2. **Shallow Copy of an Object**
      1. const person = {name: "Tony", age: 130};
      2. const personShallowCopy = {...person, gender: "male"};
10. **Rest Parameter:** Wherever you are expecting a list of values in a function, you can pass in three dots and a name as the parameters, which will merge the values into an array with the name you gave it.
    1. **Example:**

const **add** = (...numbers: number[]) => {

return numbers.reduce((curResult, curValue) => {

return curResult + curValue;

}, 0);

};

const **addedNumbers** = add(5, 10, 2, 3.7);

console.log(**addedNumbers**);

1. **Array & Object Destructuring**: Copies elements out of an array or object and stores them in variables.
   1. const hobbies = ['Sports', 'Cooking'];
   2. const [hobby1, hobby2, ...remainingHobbies] = hobbies;
   3. const person = {firstName: 'Max', age: 30};
   4. const { firstName: userName, age } = person;